Hindroid

This repository contains a mimic implementation and future implementation plan of the Hindroid paper (DO>>> I:10.1145/3097983.3098026).

Project	build Status
Data Ingesting	In progress
Feature Extraction	Not Started
ML Deployment	Not Started

What Is Hindroid

The main task of Hindroid is to use machine learning, typically Graph Neural Network, to classify Android Apps as benign or malicious. Hindroid is designed to be an intelligent Android malware detection system based on structured heterogeneous information network.

What is the Data

APK and Smali

The paper uses a static analysis method to identify malware, extracting source code from .apk files of apps. Because of reversibility of .apk files, we will decompile .apk files to Smali Code with ApkTool. We then use technique similar to Natural Language Processing to perform feature extraction outputting corresponding features, in particular, Nodes and Edges of the network.

The paper is mainly targeting on API calls in smali code. API, Application Programming Interfaces, is an interface or communication protocol between parts of a computer program intended to simplify the implementation and maintenance of software. API calls are used by Android apps in order to access operating system functionality and system resources. API calls grant possibility to apps access asking system permission to perform low level system actions like sending HTTP requests to an unknown server.

Example

API calls via Smali

```
invoke-virtual {v2, v3}, Ljava/lang/Runtime;->exec(Ljava/lang/String ;)
Ljava / lang / Process ;
```

```
invoke-virtual {v2}, Ljava / lang / Process;->getInputStream () Ljava / io
/ InputStream
```

Data Design & Collection

Abstract

The data we use in replication of paper will consist of:

- Benign Android Application from APKPure
- Malicious Android Application from our private source.

The benign apps are from an online app platform (like Playstore) APKPure. The reason we use APKPure instead of Google Playstore that APKPure is more scraping-friendly than Playstore: Playstore requires a google account to purchase free app. We can use sitemap of APKPure to sample our benign apps. More importantly, APKPure is an apk recommending site which consists app pre-census step by editors. It can reduce the possibility to get malicious app in our benign app samples. The malicious Android Application are from our private source because of to avoid the data be used in malicious way.

With the Benign sample and the Malicious sample, we have both positive and negative labels in our classification task, then we will perform ML algorithms for binary classification.

While portions focused on learning graph techniques will also use examples from other languages (for example, python and java source code). Under folder utils, building utility functions to download apk and transfer apks smali code with python

Pros

- Using Smali as our data is appropriate with following reasons:
    - perform static analysis is a novel and secure way to perform malware detection. Rather than traditional detection on apks by running in a virtual machine or actual machine, it will not execute the apks. In this way, we can prevent the malware to actual damage our personal devices while we do malware detection.
    - perform static analysis is an efficient way to process large task when we want to perform a mass malware detection over apps, not only in personal use but also in business use. Rather than detecting the malware by running the file, we scan through the code.
    - perform static analysis is more robust with iteration. Iteration by feeding in new data and tuning parameter, the classification task will follow the trend of malware and detect them precisely.
- Using APKPure as our benign data is appropriate with following reasons:
    - APKPure is a secondary app store rather than Playstore, which has significantly less census on app release. Thus, APKPure's samples are more trust-worthy and can be good positive samples.
    - APKPure is scratching-friendly. Compared to Playstore, which requires a google account to download and purchase apps, APKPure does not require an accont to download apks. Moreover, APKPure provides a sitemap on the robots.txt. We can use the sitemap to easier sample our dataset.
- The benign Android Application and Malware samples are a good match to solve our classification task. As mentioned above, APKPure
    - With balanced of positive and negative samples of apk, we can build a robust classifier to identify malware and benign apps.

Cons

- Limitation of Benign Sample
    - Although APKPure is more trust-worthy than Google Playstore, it is still questionable that every app in APKPure is benign. If a large amount of our positive samples are negative, our classifier will be less robust even invalidated. We must aware the shortcoming that not every app in APKPure is benign.
    - Since we can only download free app from APKPure, there is a big limitation of our data design: we cannot access the paid apps, which is far away from our real world scenario. Despite the low malware possibility of paid apps, we cannot neglect the sample of paid apps.
- Limitation of Malicious Sample
    - The apps from APKPure is updated over time, but our malicious sample is from historical database. There is a time gap between our Benign sample and Malicious sample, and it is not easy to keep malicious sample updated.
    - The malicious sample is much less than the benign sample. It is not easy to make two sample balanced.
- Limitation of Only Detecting API calls
    - Our paper only targets on API calls, there exit malicious apps contain non-suspect API calls, which cannot be detected by our classifier. Also, the paper neglect to analysis the relationship between each method and class.
    - The repeat use of a specific API call will not feed in to the feature extraction of the paper, which will lead an inaccuracy of classifier.

Past Efforts

- Traditional Approach
    - The traditional approach of malware detection or security threats is to scan the signature of the apps compares to the database of identified malicious apps. This approach is harder to iterate because it requires to keep update the malware database.
- Dynamic Analysis
    - Others using dynamic analysis to perform malware detection. Because this method requires an active virtual machine to run the apps, it may have security concern and it is more computationally heavy.
- Static Analysis
    - Rather than extracting API calls using a structured heterogeneous information network, some constructed similarities between apps with ML to identify malware.

Data Ingestion Process

Data Accessability

- Data Original
    - Benign Android Application from APKPure
    - Malicious Android Application from our private source.
- Legal Issues
    - According to APKPure's Term of Use

      ```
      Note: APKPure.com is NOT associated or affiliated with Google, Google
      Play or Android in any way. Android is a trademark of Google Inc. All
      the apps & games are property and trademark of their respective
      developer or publisher and for HOME or PERSONAL use ONLY. Please be
      aware that APKPure.com ONLY SHARE THE ORIGINAL APK FILE FOR FREE APPS.
      ALL THE APK FILE IS THE SAME AS IN GOOGLE PLAY WITHOUT ANY CHEAT,
      UNLIMITED GOLD PATCH OR ANY OTHER MODIFICATIONS.
      ```

      it specifies APKPure's data is only for personal use. Since our project is a personal capstone project without commercial purpose. We are free of legal Issues in data use.

    - According to APKPure's robots.txt, sitemap.xml is obtained for scraping use. Thus, we are free of violation of scraping rule.

Data Privacy

*subject to change

- According to APKPure's Privacy Policy. If necessary, we will provide our privacy information as policy requests.
- For data we collected, since it is public by APKPure, we are free of privacy concern. Regardlessly, we will still anonymise our data by following steps:
    - anonymise apk url with sha256 encryption.
    - anonymise app name with two-way hash function.
    - anonymise apk file names ,if necessary, with sha256 encryption.
    - anonymise apk developer with two-way hash function.

- anonymise apk signature ,if necessary, with sha256 encryption.
- anonymise apk category with two-way hash function.

Data Schemas

- Since we need to feed in data into a ML pipeline to make classification, we need preprocess our data, storing as a designed Data Schema like following form:

  ```
  data/
  |-- plagueinc/
  |    |-- plagueinc.apk
  |    |-- plagueinc/
  |    |    |-- AndroidManifest.xml
  |    |    |-- smali*/
  |-- instagram/
  |    |-- instagram.apk
  |    |-- instagram/
  |    |    |-- AndroidManifest.xml
  |    |    |-- smali*/
  ```

  Since apks are fairly large, and we are interested in the API call of every app. We may only keep the file AndroidManifest.xml and smali folders. For each app, after extraction of smali, we will delete the .apk file

- For each app, we will create an overall metadata.csv to store their feature according their corresponding sitemap.

  The metadata will consist following columns:

  - `loc`: the url of specific app
  - `lastmod`: the datetime of the last update of the app
  - `changefreq`: check for update frequency
  - `priority`: the priority group of the app
  - `sitemap_url`: the url in sitemap.xml

  Metadata is a map of what we will sample according to, we can do different sampling with the matadata.

Future Plan

We plan to add following features (subject to change) to the sample of metadata in feature extraction section:

- API call adjacency matrices
- developer info of specific app
- developer signiture
- name of the app
- first category (e.g. Game) of the app
- secont category (e.g. Game type) of the app
- etc.

Data Ingestion Pipeline

Data Sampling

get the list of apks url to download from `sitemap.xml`

- ✓ Initialize `metadata.csv` from `sitemap.xml`

  Initialize a metadata gives us a hint what data to sample:

- ✓ Naive sampling

  random sample same amount of apks from APKPure to the malware sample.

  usage

  sampling 1000 benigned apks

  ```
  import json
  sys.path.append('./utils')
  import utils
  import pandas as pd
  cfg = json.load(open('./sitemap.json'))
  utils.create_sitemap_df(**cfg) #Create a sitemap dataframe with
  corresponding info.
  metadata = pd.read_csv('./data/metadata/metadata.csv')
  metadata.sample(1000)
  urls = metadata.loc
  ```

- ☐ Category Sampling *will be inplement after feature extraction
  - sampling same number of apks according to corresponding category from APKPure with the malware sample.
  - First sample a smaller set from sitemap, then fetch the category of each apps by requesting apps' links. With each category get the even matched links to sample.

    usage

    ```
    ###TODO
    ```

- ☐ Future Sample Methods Coming Soon...
  - update after observation of first two sampling methods.

Data Downloading

- ✓ Given a `app-url.json` to execute download.

  For example, to download `facebook` and `Plague Inc.` apps to `./data` directory the `app-url.json` may look like:

```
{
"data_dir": "./data",
"urls": [
    "https://apkpure.com/plague-inc/com.miniclip.plagueinc",
    "https://apkpure.com/instagram/com.instagram.android"
    ],
"verbose": 1
}
```

usage

```
import json
import re
sys.path.append('./utils')
import utils
cfg = json.load(open('./demo/app-url.json'))
urls, fp = cfg['urls'], cfg['data_dir']
for url in urls:
  app = re.findall(r'https:\/\/apkpure.com\/(.*?)\/', url)[0]
  utils.download_app(url, fp, app)
```

Converting apks to smali

- ✓ APK -> Smali using apktool

  check the documentation of APKTool

Fetching and Storing Data

The complete pipeline of getting both metadata and downloading apk and decompose them into data schemas.

Demo Notebook

- ✓ fetching data consists downloading apk and decompose them into data schemas.

  usage

```
import json
sys.path.append('./utils')
import utils
cfg = json.load(open('./demo/app-url.json'))
utils.get_data(**cfg)
>>> fetched ./data/plague-inc/plague-inc.apk, start decoding
>>> I: Using Apktool 2.4.1 on plague-inc.apk
>>> I: Loading resource table...
>>> I: Decoding AndroidManifest.xml with resources...
>>> I: Loading resource table from file:
/Users/syeehyn/Library/apktool/framework/1.apk
```

```
>>> I: Regular manifest package...
>>> I: Decoding file-resources...
>>> I: Decoding values */* XMLs...
>>> I: Baksmaling classes.dex...
>>> I: Copying assets and libs...
>>> I: Copying unknown files...
>>> I: Copying original files...

fetched ./data/instagram/instagram.apk, start decoding
>>> I: Using Apktool 2.4.1 on instagram.apk
>>> I: Loading resource table...
>>> I: Decoding AndroidManifest.xml with resources...
>>> I: Loading resource table from file:
/Users/syeehyn/Library/apktool/framework/1.apk
>>> I: Regular manifest package...
>>> I: Decoding file-resources...
>>> I: Decoding values */* XMLs...
>>> I: Baksmaling classes.dex...
>>> I: Baksmaling classes2.dex...
>>> I: Baksmaling classes3.dex...
>>> I: Baksmaling classes4.dex...
>>> I: Copying assets and libs...
>>> I: Copying unknown files...
>>> I: Copying original files...
```

- [x] fetching the sitemap DataFrame

  usage

  ```
  import json
  sys.path.append('./utils')
  import utils
  utils.setup_env()
  cfg = json.load(open('./demo/sitemap.json'))
  utils.create_sitemap_df(**cfg)
  ```

Feature Extraction

- [ ] Smali -> Graph Embedding & Feature Extraction with ML framework.
- [ ] etc.

Machine Learning

- [ ] planning

Prerequisite

- ApkTool to decompile an Android Apk to Smali code

Installation

APKTool and Java dependency is preinstalled in repo

The directory of ApkTool is

```
./utils/apktool
./utils/apktool.jar
```

The directory of Java is

```
./utils/jre1.8.0_241/
```

if you are using bash

```
echo $"export PATH=$PATH:<REPO_DIR>/utils" >> ~/.bash_profile
echo $"export PATH=$PATH:<REPO_DIR>/utils/jre1.8.0_241/bin" >>
~/.bash_profile
```

if you are using zsh

```
echo $"export PATH=$PATH:<REPO_DIR>/utils" >> ~/.zshrc
echo $"export PATH=$PATH:<REPO_DIR>/utils/jre1.8.0_241/bin" >>
~/.zshrc
```

- networkX
- DockerFile of the environment

References

References are found both in the weekly readings, as well as in references. These will be update throughout the quarter.

HinDroid paper on Malware detection.

- Malware Background
    - Computer Viruses and Malware by Aycock, John. Available for pdf download on campus networks or VPN.
    - Slides for the Malware and Cybercrime lecture of CSE 127 at UCSD.
    - A reference sheet for decompiling Android applications to Smali Code.

- Graph Basics

...

- Graph Techniques in Machine Learning
  - A (graduate) survey course at Michigan on Graph Mining.
- Machine Learning on Source Code
  - A collection of papers and references exploring understanding source code with machine learning.