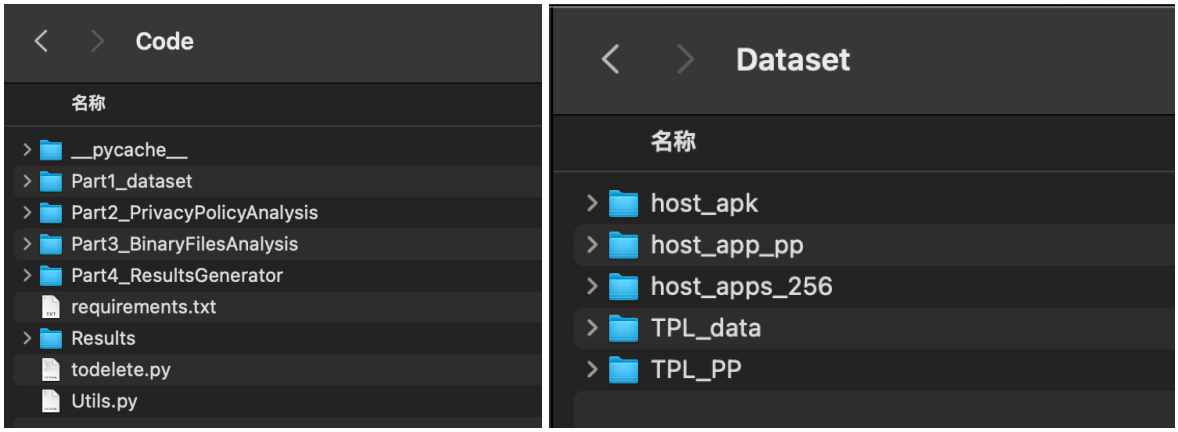
# ATPChecker

The source code ("https://doi.org/10.5281/zenodo.8248630" -> ./Code.zip) and the data set ( "https://doi.org/10.5281/zenodo.8248630" -> Dataset.zip.*) for ATPChecker are given in the repository (10.5281/zenodo.8015961). The users can also download the code and dataset from OneDrive . After downloading the source file, please unzip them in the same folder.

The users are suggested to use IntelliJ IDEA and Pycharm to load the project and use the tool.

## Step-1. Preparation

After downloading the **Code.zip** ("https://doi.org/10.5281/zenodo.8248630" -> ./Code.zip) and Dataset.zip.* ( "https://doi.org/10.5281/zenodo.8248630" -> Dataset.zip.*), and unzipping them, the file structure should be as follows:




- To unzip the dataset, the users are suggestions to use the following scripts in terminal to unzip the files:

```
cat Dataset.zip.* > Dataset.zip
unzip Dataset.zip
```

- To use ATPChecker, the users' environment should stisfy the following requirements:

### Context in the Dataset.zip

1. *./Dataset/TPL_data*: The folder contains binary files of the collected third-party libraries.
2. *./Dataset/TPL_pp*: The folder contains the privacy policies of the collected third-party libraries.
3. *./Dataset/host_apk*: The folder provides the binary files of the host apps.
4. *./Dataset/host_app_pp*: The folder provides the privacy policies of host apps.
5. *./Dataset/host_apps_256/APKs*: The folder provides the binary files of host apps for RQ4.
6. *./Dataset/host_apps_256/PrivacyPolicy*: The folder provides the privacy policies of host apps for RQ4.

## Context in the Code.zip

1. *./Code/DownloadAPK*: The folder contains necessary scripts to assist users in downloading Android software and their corresponding privacy policies on their own.
2. *./Code/Part1_dataset*: The scripts in this folder help to statistically analyze and display basic information about various types of data in the dataset.
3. *./Code/Part2_PrivacyPolicyAnalysis*: The folder provides necessary scripts for preprocessing and analyzing the privacy policies of third-party libraries and host apps.
4. *./Code/Part3_BinaryFilesAnalysis*: The folder contains necessary scripts for analyzing binary files of third-party libraries and host apps.
5. *./Code/Part4_ResultsGenerator*: The folder provides necessary scripts for reproducing results in the manuscript.
6. *./Code/Results*: The folder provide intermediate results for reproducing results in the manuscript.

**For Python environment:**

The authors can use the following script to create the virtual environmrnt to use the tool:

```
conda create -n atpchecker python==3.8
conda activate atpchecker
pip install matplotlib~=2.2.3
pip install hanlp~=2.1.0b17
pip install phrasetree~=0.0.8
pip install nltk~=3.6.7
pip install html2text~=2020.1.16
pip install roman~=3.3
pip install langid~=1.1.6
pip install beautifulsoup4~=4.10.0
```

Afterwards, the users can config the virtual environment *atpchecker* in Pycharm to use the tool.

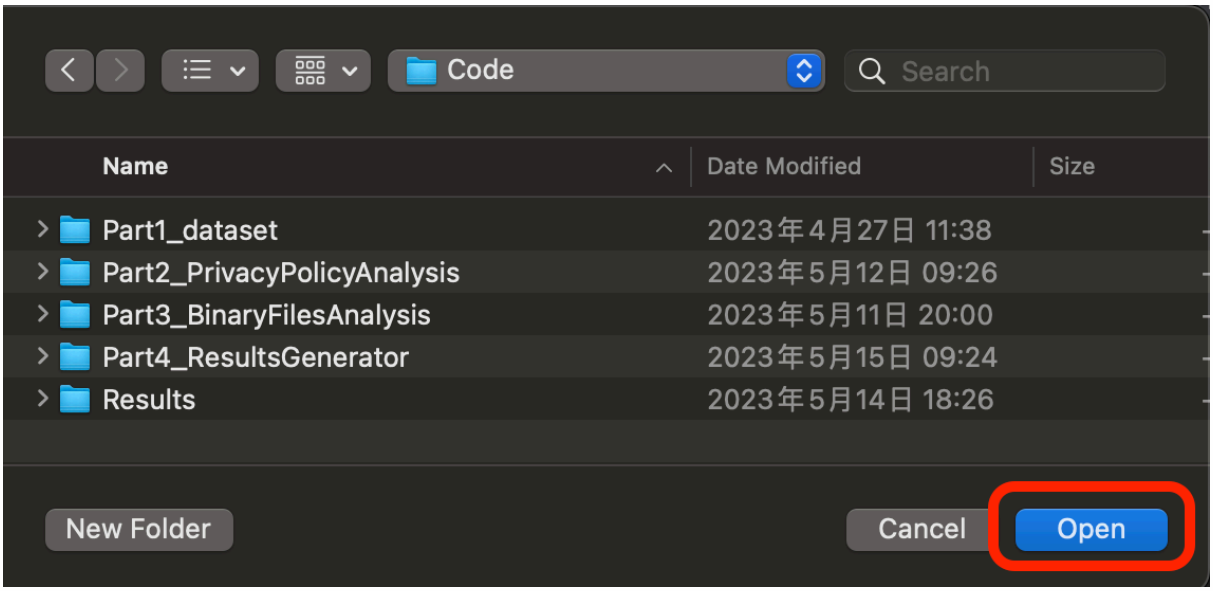
**For Java environment:**

The users are suggested to use JDK-1.8 to load the tool.

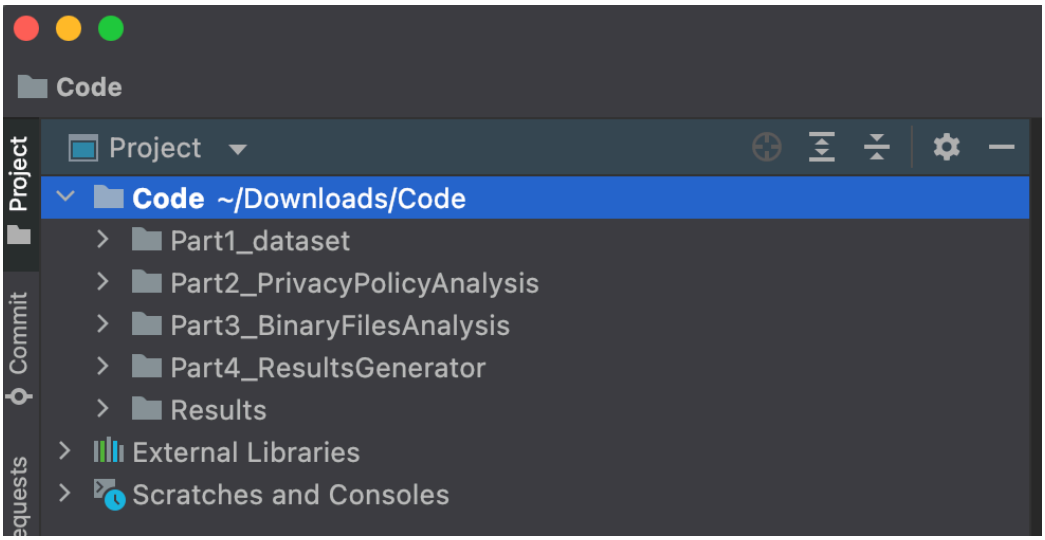
## Step0. Load the project

The users can use Pycharm to open the project directly:

*Pycharm -> Files -> open -> the_path_to_code -> open*

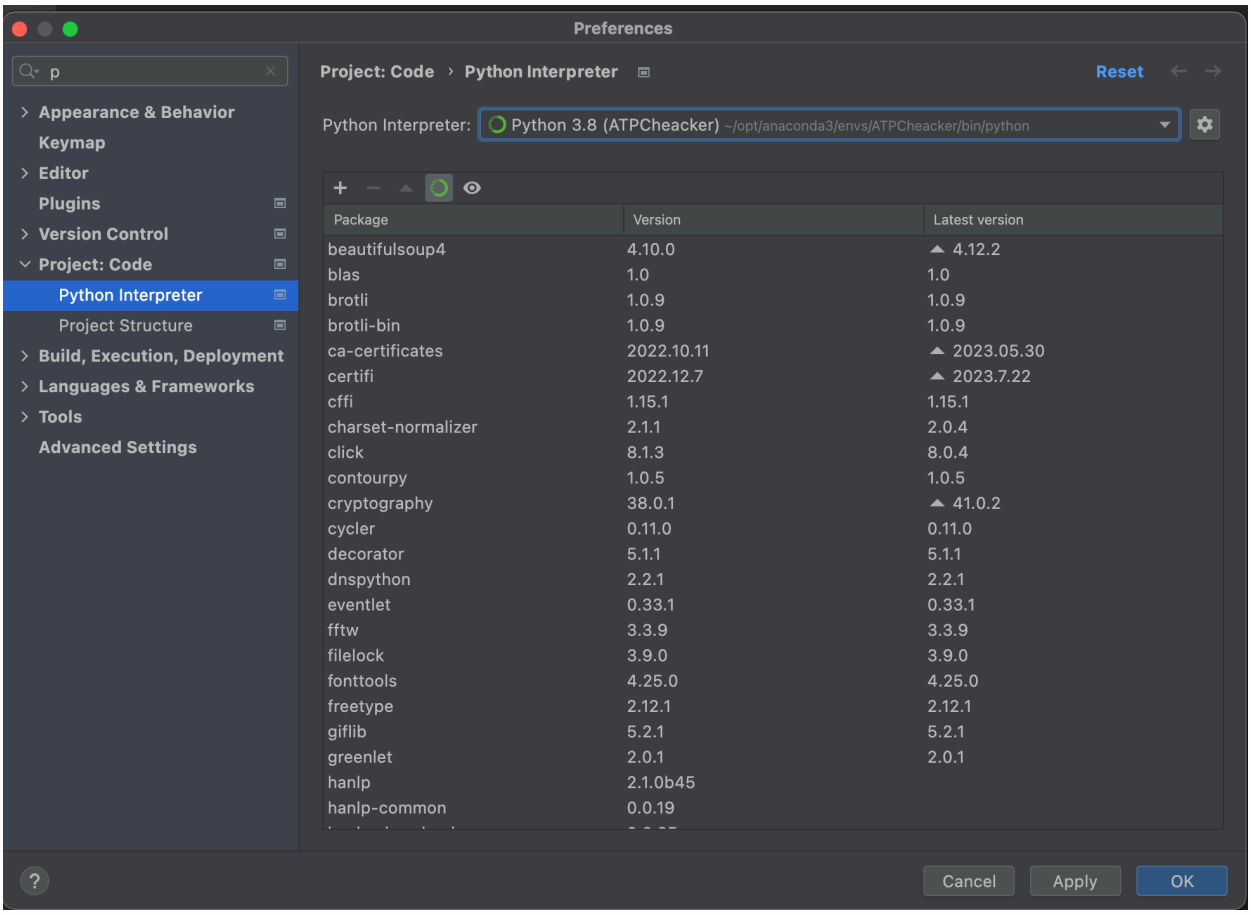



The project structure should be like the following in pycharm:




Then, the users should configure the virutal environment *atpchecker* by:

*Pycharm -> Settings... -> Project:Code -> Python Interpreter -> atpchecker -> OK*




# Step1. Get Dataset Information

### 1.1 Get information of TPL list

The users can run "*./Code/Part1_dataset/Step1_1_get_TPL_list_info.py*" to get the information of TPL list. Before that, please replace the path to ***TPL_PP* folder** in line_48 with the path you download the dataset.

```
47 if __name__ == '__main__':
48     TPL_list_path = "/ATPChecker/dataset/TPL_PP"
49     main(TPL_list_path)
50
```

Then, the users can get the information of the TPL list used in our dataset.

```
Run: Step1_1_get_TPL_list_info
    "Socialize",
    "swarm",
    "TikTok open SDK",
    "Twitter API ME",
    "Twitter Kit",
    "Twitter4j",
    "VKontakte SDK",
    "WeChat",
 ==============================
The number of TPLs in ad_networks: 141
The number of TPLs in development_tools: 292
The number of TPLs in social_libraries: 25

Total number of TPLs: 458
Copyright of the list is reserved by AppBrain.

Process finished with exit code 0
```

### 1.2 Get information of TPLs' privacy policy files

The users can run "*./Code/Part1_dataset/Step1_2_get_TPL_privacy_policy_files.py*" to get the information of TPLs' privacy policies in our dataset. Before that, please replace the path to ***TPL_PP* folder** in line 54 to the path you downloaded the dataset.

```
53 if __name__ == '__main__':
54     TPL_pp_folder = "/ATPChecker/dataset/TPL_PP"
55     main()
56
```

Then, the users can get the information of TPLs' privacy policies in our dataset.

```
Run: Step1_2_get_TPL_privacy_policy_files
    asmack
    Digits for Android
    JTwitter
    Papaya Social
    Playhaven
    ScoreLoop
    Smack API
    Socialize
    swarm
    Twitter4j
 ==============================
    ad_networks: 120
    development_tools: 112
    social_libraries: 15
The dataset contains 247 privacy policies

Process finished with exit code 0
```

### 1.3 Get the information of TPLs' binary files

The users can run "*./Code/Part1_dataset/Step1_3_get_TPL_binaryfiles_info.py*" to get the information of TPLs' binary files in our dataset. Before that, please replace the path to *TPL_data* folder in line 47 to the path you downloaded the dataset.

```
Step1_3_get_TPL_binaryfiles_info.py

if __name__ == '__main__':
    TPL_binaryfiles_folder = '/ATPChecker/dataset/TPL_data'
    main(TPL_binaryfiles_folder)
```

Then, the users can get the information of TPLs' binary files in our dataset.

```
Run: Step1_3_get_TPL_binaryfiles_info
social_libraries >> com.google.guava >> guava-30.1.1-jre.jar
social_libraries >> com.magnet.mmx.ext >> mmx-asmack-android-8-4.0.7p3.jar
social_libraries >> com.socialize >> loopy-3.1.7.aar
social_libraries >> com.tencent.mm.opensdk >> wechat-sdk-android-without-mta-6.6.4.aar
social_libraries >> org.jetbrains.kotlin >> kotlin-android-extensions-runtime-1.2.40.jar
social_libraries >> org.slf4j >> slf4j-api-1.6.5.jar
social_libraries >> org.springframework.social >> spring-social-core-1.1.2.RELEASE.jar
social_libraries >> org.twitter4j >> twitter4j-core-2.1.10.jar
 ==============================
The dataset contains 187 distinct TPL binary files and includes 100 *.jar files and 87 *.aar files:
    ad_networks: 45
    development_tool: 132
    social_libraries: 10

Process finished with exit code 0
```

### 1.4 Get the information of host apps

The users can run "*./Code/Part1_dataset/Step1_4_get_host_app_info.py*" to get the information of TPLs' binary files in our dataset. Before that, please replace the path to *host_apk* folder in line 31 with the path you downloaded the dataset.

```
Step1_4_get_host_app_info.py
27      print('The number of distinct host apps in our datas
28
29
30  if __name__ == '__main__':
31      host_apk_folder = '/ATPChecker/dataset/host_apk'
32      main(host_apk_folder)
```

Then, the users can get the information of host apps' files in our dataset.

```
Step1_4_get_host_app_info
The number of distinct host apps in our dataset: 641

Process finished with exit code 0
```

## Step2. Privacy policy analysis.

### 2.1 Preprocess TPL privacy policies

The users can run "./*Code/Part2_PrivacyPolicyAnalysis/Step2_1_preprocess_TPL_privacy_policy.py*" to preprocess the TPLs' raw privacy policies. This step will transform the raw privacy policy files, which are in the format of *html* files, into plain text files. Before that, please replace the path to ***tpl_pp* folder** in line_39 to the path you downloaded the dataset and replace the variable *save_folder* in line_40 to the path you want to save the preprocessed privacy policies.

Step2_1_preprocess_TPL_privacy_policy.py

```
33 if __name__ == '__main__':
34     '''
35     The TPL privacy policy should be saved under th
36     The pre-processed privacy policies will be save
37     '''
38
39     tpl_pp_folder = '/ATPChecker/dataset/TPL_PP'
40     save_folder = '../Results/preprocessed_pp/'
41     main(tpl_pp_folder, save_folder)
42
```

#### 2.1.1 Get the basic infomation of preprocessed privacy policies.

The users can run "./*Code/Part2_PrivacyPolicyAnalysis/Step2_1_1_get_basic_infomation.py*" to get the basic information of preprocessed TPLs' privacy policies, including the sentence number in the privacy policies, the number of WWWH sentences, etc. This function reproduces the results presented in Section.IV.B.**Results**.***TPL privacy policies analysis***. Before that, please replace the path to **preprocessed TPL's privacy policies** (which are generated by Step 2.1) in line_110.

Step2_1_1_get_basic_infomation.py

```
108
109 if __name__ == '__main__':
110     TPL_pp_folder = "../Results/preprocessed_pp/"
111     PRINT_FLAG = False  # flag to print detail report
112     main(TPL_pp_folder, PRINT_FLAG)
113
```

The users can get basic informations as is presented in Section IV.B.***TPL privacy policies analysis***.

```
Run: Step2_1_1_get_basic_infomation
/Users/zhaokaifa_imac/opt/anaconda3/envs/py36/bin/python /Volumes/Samsung8702/ATPCh
==========

Number of sentences in all privacy policies: 91155
Number of sentences starting with 'who, why, when, whether, what, how' 15270
Number of sentences with sharing and collection (SAC) words: 52523
Number of sentences consider as SAC: 37253
Number of sentences ambiguously declaring data access behavior: 5205

Process finished with exit code 0
```

**2.2 Analyze TPL's privacy policies.**

The users can run "*./Code/Part2_PrivacyPolicyAnalysis/Step2_2_TPL_PP_analysis.py*" to analyze the preprocessed TPLs' privacy policies. Before that, please replace the path to **preprocessed TPL's privacy policies** in line_26 (variable *TPL_PP_preprocessed_folder*) and the variable *TPL_PP_analysis_results* to the path you want to save the analyzed results.

```
Step2_2_TPL_PP_analysis.py
24
25 if __name__ == '__main__':
26     TPL_PP_preprocessed_folder = '../Results/preprocessed_pp/'
27     TPL_PP_analysis_results = '../Results/TPL_pp_analysis_results'
28     main(TPL_PP_preprocessed_folder, TPL_PP_analysis_results)
29
```

**2.3 Preprocess host apps' privacy policies.**

The users can run "*./Code/Part2_PrivacyPolicyAnalysis/Step2_3_preprocess_hostapp_pp.py*"to preprocess the host apps' privacy policies. Before that, please replace the path to **host apps' privacy policies** with the path you downloaded the data set (variable *host_app_pp_folder*) in line_30 and replace the variable *pp_save_folder* you want to save the preprocessed privacy policies.

```
Step2_3_preprocess_hostapp_pp.py
29 if __name__ == '__main__':
30     host_app_pp_folder = '/dataset/host_app_pp'
31     pp_save_folder = '../Results/preprocessed_hostapp_pp/'
32     main(host_app_pp_folder, pp_save_folder)
33
```

**2.4 Analyze host apps' privacy policies.**

The users can run "./*Code/Part2_PrivacyPolicyAnalysis/Step2_4_host_app_PP_analysis.py*" to analyze the host apps' privacy policies. Before that, please replace the **path to host apps' privacy policies** with the path of preprocessed host apps' privacy policies (variable *host_app_pp_preprocessed_folder*) and replace the variable *host_app_pp_analysis_results* you want to save the analyze results.

```
Step2_4_host_app_PP_analysis.py
27 if __name__ == '__main__':
28     host_app_pp_preprocessed_folder = '../Results/preprocessed_hostapp_pp/'
29     host_app_pp_analysis_results = '../Results/hostapp_pp_analysis_results'
30     main(host_app_pp_preprocessed_folder, host_app_pp_analysis_results)
```

**2.5 Anlyze host apps' privacy policies for RQ4.**

The users can run "./*Code/Part2_PrivacyPolicyAnalysis/Step3_256app_privacy_policy_analysis.py*" to analyze the host apps' privacy policies. Before that, please replace the **path to host apps' privacy policies** in line 27 (variable *app_privacy_policy_path*) with the path of preprocessed host apps' privacy policies, i.e., *'./Dataset/host_apps_256/PrivacyPolicy'*, and replace the variable *host_app_pp_analysis_results* you want to save the analyze results.

```
26 if __name__ == '__main__':
27     host_app_pp_preprocessed_folder = './Dataset/host_apps_256/PrivacyPolicy'
28     host_app_pp_analysis_results = '../Results/hostapp_256_pp_analysis_results'
29     main(host_app_pp_preprocessed_folder, host_app_pp_analysis_results)
30
```

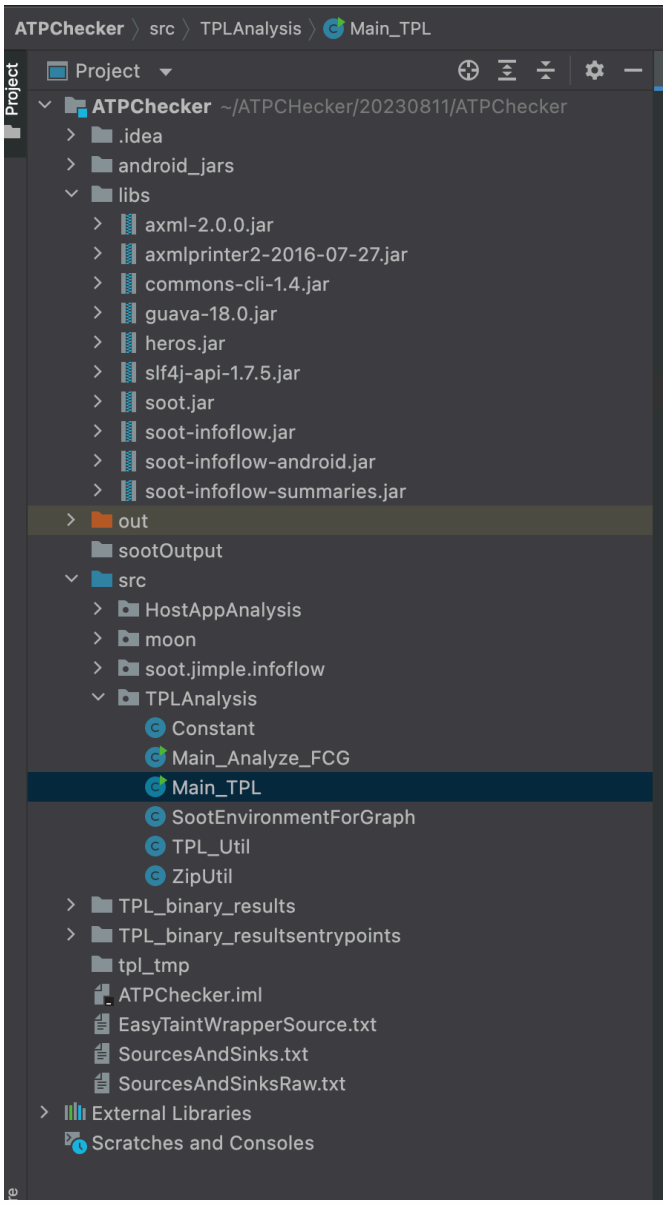
## Step3. Binary files analysis.

**3.0 Load the project using IDEA.**

The users can use IDEA to load the project directly:

*IDEA -> File -> Open -> ./Code/Part3_BinaryFilesAnalysis/ATPChecker -> Open*

After loading the project using IDEA, the project dependencies will be loaded accordingly. The project structure should be like:




We have provided the lasted *android_jar* in "./*ATPChecker/android_jars*" to test the functionality of ATPChecker. To reproduce the results in our manuscript, please download all android_jar versions from https://github.com/ATPChecker/ATPChecker_jars and put them in the "./*ATPChecker/android_jars*".

### 3.1 Evaluate ATPChecker's analysis capacity.

The users can run "./*ATPChecker/src/TPLAnalysis/Main_Analyze_FCG.java*" in IDEA to evaluate the capacity of ATPChcker to analyze TPLs. Before that, please replace the path to TPL's binary files in line_18 to the path you downloaded the TPL's binary files (variable *root_tpl_folder* to the path "**/Dataset/TPL_data**") and replace the variable *fcg_save_folder* to the path you want to save the analyze results. This step is to evaluate the capacity of ATPChcker to analyze TPLs' binary files and the results generated this step will be used to reproduce results in Figure.2.

```java
import java.util.List;

public class Main_Analyze_FCG {
    private static String tmp_folder = "tpl_tmp";
    private static String root_tpl_folder = "/ATPChecker/dataset/TPL_data";
    private static String fcg_save_folder = "./FCG_Compare";


    public static void main(String[] args) throws Exception {
        File tpl_cate_file = new File(root_tpl_folder);
        File[] tpl_cate_list = tpl_cate_file.listFiles();
        String output_path = "test";
        checkFolder(fcg_save_folder);

        // 写文件
        String file_name = "fcg_statistic.csv";
```

- Two results will be generated. The results in "./FCG_Compare" gives the extrated function call graph (FCG) analyzed by raw soot and our optimized mechanism. The results in "fcg_statistic.csv" give the number of nodes and edges of FCG. Those two results will be used in **RQ_2.3 Reproduce Figure 2** to reproduce Figure 2 in our manuscript. Please notice that

* Please note that the analysis capabilities may vary for different APKs due to differences in computer CPU computing power and memory. We also provide intermediate results (FCG files in './ATPChecker/FCG_Compare' and './ATPChecker/fcg_statistic.csv') obtained from running on our devices to reproduce the results presented in the manuscript.

### 3.2 Analyze TPLs' binary files.

The users can run "*./ATPChecker/src/TPLAnalysis/Main_TPL.java*" in IDEA to analyze TPLs' binary files. Before that, please replace the path to TPLs' binary files in line 30 with the path you downloaded the **TPL's binary files** (variable *root_tpl_folder* to the path "/Dataset/TPL_data") and replace the variable *root_results_folder* in line 26 to the path you want to save the analyze results.

```java
22  import static TPLAnalysis.TPL_Util.transAar2Jar;
23  import static TPLAnalysis.ZipUtil.deleteDir;
24
25  public class Main_TPL {
26      private static String root_results_folder = "./TPL_binary_results";
27      private static String tmp_folder = "./tpl_tmp";
28
29      public static void main(String[] args) {
30          String root_tpl_folder = "/ATPChecker/dataset/TPL_data";
31          File tpl_cate_file = new File(root_tpl_folder);
```

- This script analyzes the TPLs' user data access behavior in their code. The results will be saved on *root_results_folder*. In the folder, each TPL will generate a results file that contains the statement of the TPL's data access behavior and the data type inside. For example, for the TPL *androidx.appcompat:*

```
androidx.appcompat.txt
1  stmt:   $r4 = virtualinvoke $r3.<android.location.LocationManager: android.location.Location
   getLastKnownLocation(java.lang.String)>(r1)
2        data   android.location.LocationManager getLastKnownLocation
3                results::  ImmediateBox($r4) in return $r4
4
5  stmt:   $d0 = virtualinvoke r3.<android.location.Location: double getLatitude()>()
6        data   android.location.Location getLatitude
7                results::  ImmediateBox($d0) in virtualinvoke r2.<androidx.appcompat.app.TwilightCalculator:
                 void calculateTwilight(long,double,double)>($l1, $d0, $d1)
8                results::  ImmediateBox(d34) in d35 = d34 * 0.01745329238474369
```

  - Line1: "*stmt: ...*" denotes the analyzed traces of the TPL's data access to ***location***. The line 2 gives the data type.
- These results will be used in **Step4. RQ_2.2 TPL compliance analysis**.

* Please note that the analysis capabilities may vary for different TPLs due to differences in computer CPU computing power and memory. We also provide intermediate results ('./Code/Results/TPL_binary_results') obtained from running on our devices to reproduce the results presented in the manuscript.

### 3.3 Analyze host apps' binary files.

The users can run "*./ATPChecker/src/HostAppAnalysis/Main_Hostapp.java*" in IDEA to analyze host apps' binary files. Before that, please replace the path to the **host apps' binary files** to the path you downloaded the host apps' binary files (variable *app_path* to the path "/Dataset/host_apk") and replace the variable *root_save* to the path you want to save the analyze results.

```
Main_Hostapp.java
10  import static HostAppAnalysis.Hostapp_Util.*;
11
12  public class Main_Hostapp {
13      public static void main(String[] args) throws Exception {
14          String app_path = "/ATPChecker/dataset/host_apk";
15          String root_save = "host_app_binary_results";
16          check_dir(root_save);
17          File app_list = new File(app_path);
```

- This script analyzes host apps' data interaction with TPLs. The results will be save under "*root_save*" folder. In the folder, each app will generate a results file that contains the host app's data sharing behavior withTPLs, the data type and the TPL's package name. For example, for the app *com.goodbarber.inoitips_apk*:

- ```
com.goodbarber.inoitips_apk.txt
1 $r3 = staticinvoke <com.goodbarber.v2.core.data.languages.Languages: java.lang.String getEmail()>(): TPL
  :com.goodbarber
2     PI:email : $r3 = staticinvoke <com.goodbarber.v2.core.data.languages.Languages: java.lang.String
      getEmail()>()
3     TPL : com.goodbarber : ImmediateBox($r3) in $r24 = interfaceinvoke
      $r23.<com.goodbarber.v2.core.data.content.IDataManager: android.graphics.Typeface getTypeface(
      java.lang.String)>($r3)
4 $r4 = staticinvoke <com.google.android.gms.ads.identifier.AdvertisingIdClient:
  com.google.android.gms.ads.identifier.AdvertisingIdClient$Info getAdvertisingIdInfo(
  android.content.Context)>($r3): TPL :com.google.android.gms
5     PI:advertisingidClient getadvertisingidInfo : $r4 = staticinvoke <
      com.google.android.gms.ads.identifier.AdvertisingIdClient:
      com.google.android.gms.ads.identifier.AdvertisingIdClient$Info getAdvertisingIdInfo(
      android.content.Context)>($r3)
6     TPL : com.google.android.gms : JimpleLocalBox($r8) in specialinvoke
      $r8.<com.google.android.gms.ads.identifier.AdvertisingIdClient$Info: void <
      init>(java.lang.String,boolean)>($r9, $z0)
7     TPL : com.google.android.gms : ImmediateBox($r4) in specialinvoke
```
  - Line4:"*$r4 = static...*" denotes the trace that locates the data access behavior.
  - Line5: "\t*PI:adver...*" gives the personal information type and the traces that the app access the data.
  - Line6: "\tTPL:..." gives the TPL's package name and traces the data in line 5 to the TPL.
  - The results will be used in **Step4.RQ_3.1 Reproduce results in RQ_3** and **Steps4.RQ_3.2 Reproduce Figure 5**.
- Please note that the reproduced results may differ from our results because of the difference in the computer's hardware capacity. Some apps may fail due to time-out settings in our code. We evaluate the performance of our tool on an iMac-2017 with Intel Core i7, 32 GB memory. If your computer's computation capacity is better than the given configuration, maybe more apps can be analyzed; otherwise, more apps may fail.

### 3.4 Analyze host apps for RQ4

The users can run "./*ATPChecker/src/HostAppAnalysis/Main_host_app_256.java*" in IDEA to analyze host apps' binary files. Before that, please replace the path to the **host apps' binary files** to the path you downloaded the host apps 256's binary files (variable *app_path* to the path "./Dataset/host_apps_256/APKs") and replace the variable *root_save* to the path you want to save the analyze results. Please differentiate the saving paths for Steps 3.3 and 3.4.

```
12 public class Main_host_app_256 {
13     public static void main(String[] args) throws Exception {
14         String app_path = "/ATPChecker/Dataset/host_apps_256/APKs";
15         String root_save = "256host_app_binary_results";
16         check_dir(root_save);
17         File app_list = new File(app_path);
18         //
```

* Please note that the reproduced results may differ from our results because of the difference in the computer's hardware capacity. Some apps may fail due to time-out settings in our code. We evaluate the performance of our tool on an iMac-2017 with Intel Core i7, 32 GB memory. If your computer's computation capacity is better than the given configuration, maybe more apps can be analyzed; otherwise, more apps may fail.

## Step4. Results generator

### RQ_2.2 TPL compliance analysis
```

The users can run "*./Code/Part4_ResultsGenerator/RQ_2_2_TPL_compliance_analysis.py*" in Pycharm to get the information of TPLs' that compliance the regulation requirements for disclosing their data usage in their privacy policies. Before that, please 1. replace the path to the analysis **results of TPLs' binary files** generated by **Step3.2** in *line_255* and 2. replace the path to the **analysis results of TPLs' privacy policies** generated by **Step2.2** in *line_256*. The step is to reproduce the answers to RQ2.

```
253 if __name__ == '__main__':
254     tpl_name_map = get_tpl_name_map('./TPL_package_mapping/')
255     app_info_ss_tmp = get_tpl_ss_data('../Results/TPL_binary_results')
256     app_info_pp_tmp = get_tpl_pp_data('../Results/TPL_pp_analysis_results')
257     #
258     app_info_pp = {}
259     for tpl in app_info_pp_tmp:
260         if len(app_info_pp_tmp[tpl]) != 0:
261             app_info_pp[tpl] = app_info_pp_tmp[tpl]
262     app_info_ss = {}
263     for tpl in app_info_ss_tmp:
264         if len(app_info_ss_tmp[tpl]) > 0:
265             app_info_ss[tpl] = app_info_ss_tmp[tpl]
```

```
if __name__ == '__main__'
```

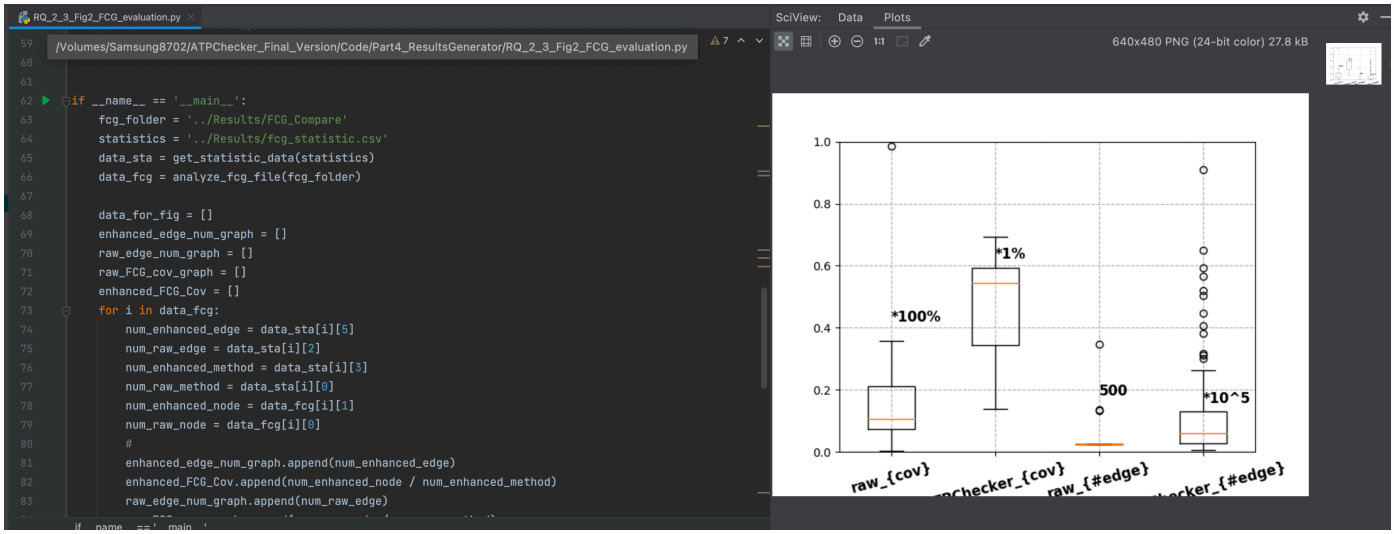
```
Run: RQ_2_2_TPL_compliance_analysis
com.noqoush.adfalcon.android missing disclose android.location.Location getLatitude in its privacy policy.
com.noqoush.adfalcon.android missing disclose android.location.Location getLongitude in its privacy policy.
com.paypal.sdk missing disclose android.telephony.TElephonyManager getDeviceId in its privacy policy.
com.paypal.sdk missing disclose android.telephony.TelephonyManager getLine1Number in its privacy policy.
com.paypal.sdk missing disclose android.net.wifi.WifiInfo getMacAddress in its privacy policy.
com.paypal.sdk missing disclose android.net.wifi.WifiInfo getSSID in its privacy policy.
com.paypal.sdk missing disclose android.telephony.TElephonyManager getSubscriberId in its privacy policy.
com.paypal.sdk missing disclose advertisingidClient getadvertisingidInfo in its privacy policy.
com.paypal.sdk missing disclose advertisingidClient getadvertisingidInfo in its privacy policy.
com.pollfish missing disclose advertisingidClient getInfo in its privacy policy.
com.unity3d.ads missing disclose android.hardware.SensorManager getDefaultSensor in its privacy policy.
io.display missing disclose advertisingidClient getadvertisingidInfo in its privacy policy.
TPLs that mis disclose data usage:
# TPLs that miss disclosing data usage 15 / 38 = 0.394737

Process finished with exit code 0
```

- The list shown in the console gives the list of TPLs. The list of TPLs indicates our tool's identification results that the TPLs do not disclose their data access behavior in privacy policies. The list is also used to generate Figure.4.
  - Take *com.inmobi.monetization/inmobi-ads* for example. We first search the keywrods, i.e., inmobi, Maven Repository (https://mvnrepository.com), and get the results page (https://mvnrepository.com/search?q=inmobi). Then, we the number of TPLs that usage the TPL, i.e., the link: https://mvnrepository.com/artifact/com.inmobi.monetization/inmobi-ads/usages. To this point, the sencond column in Figure 4 for *com.inmobi.monetization/inmobi-ads* is obtained.Then, given the information of TPLs given in the results page ( https://mvnrepository.com/artifact/com.inmobi.monetization/inmobi-ads/usages), we summarize the number of TPLs that use the TPL which uses the com.inmobi.monetization/inmobi-ads. Then, the third column in Figure 4 is obatained.
  - !!! Please note that as time progresses, the number of TPLs using the target TPL may increase.
- The output in the console (i.e., 15/38=0.3947) indicates there are 15 out of 38 TPLs miss disclosing their data usage in their privacy policies. The 38 denotes the number of TPLs identified as conducting user data access behavior in their code. The results provide the answer to RQ2..Additionally, the console also indicates which third-party libraries have not declared the personal information used by the code in their privacy policies.
  -

### RQ_2.3 Reproduce Figure 2

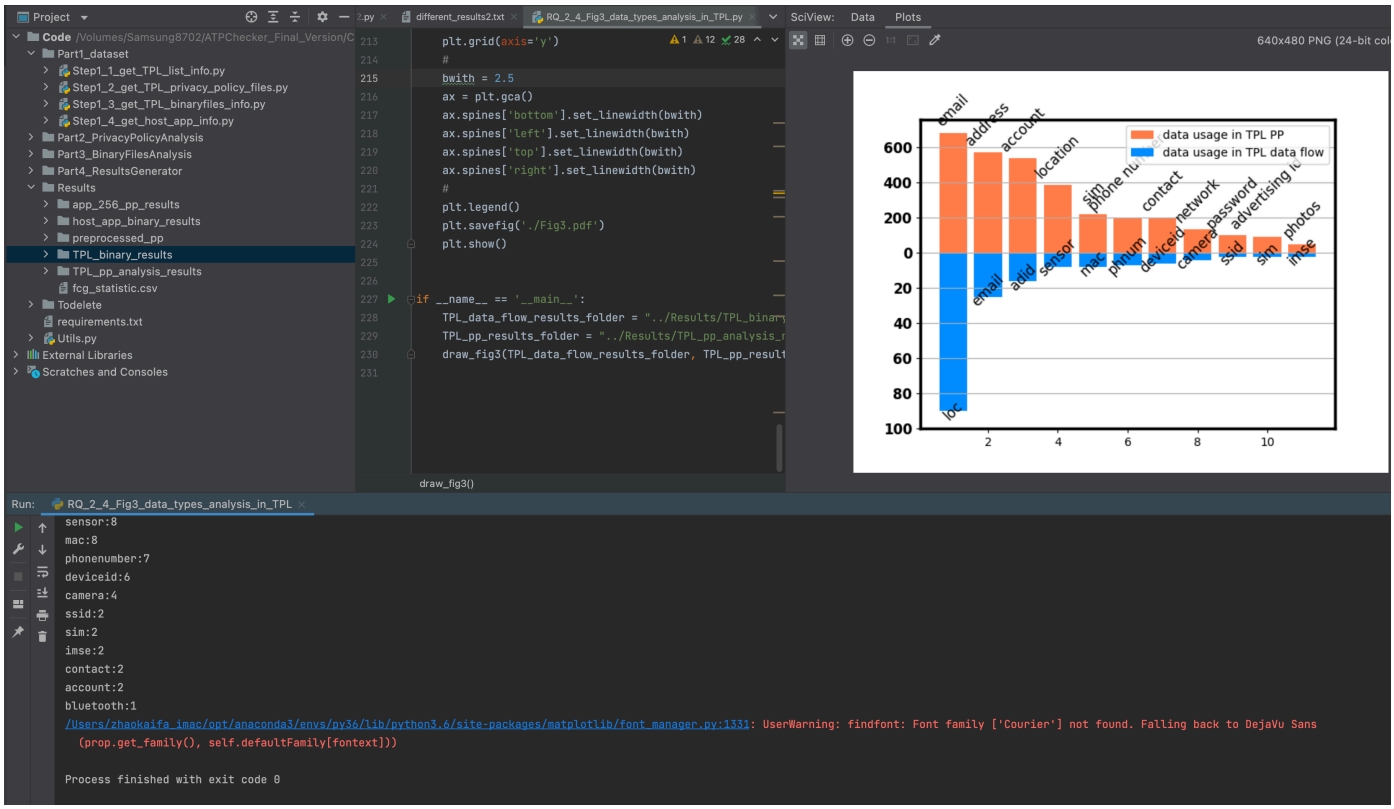
The users can run "*./Code/Part4_ResultsGenerator/RQ_2_3_Fig2_FCG_evaluation.py*" in Pycharm to reproduce Figure 2 which evaluate the capacity of ATPChcker for analyze TPLs' binary files. Before that, please replace the variable ***fcg_folder*** in line 63 and ***statistics*** in line 64 to the **results folder generated by Step.3.1**.




- The figure corresponds to Figure 2 in our manuscript.

### RQ_2.4 Reproduce Figure 3

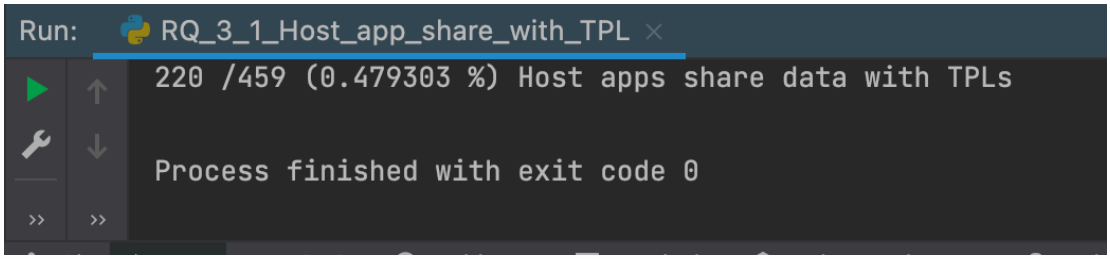
The users can run "*./Code/Part4_ResultsGenerator/RQ_2_4_Fig3_data_types_analysis_in_TPL.py*" to reproduce Figure 3. Before that, please replace the variable *TPL_data_flow_results_folder* in line 228 with the path you save the **TPLs' binary file analysis results**, which are generated by ***Step3.2***, and the variable *TPL_pp_results_folder* in line 229 to the **TPLs' privacy policy analysis results**, which are generated by ***Step 2.2***.



- The information printed in the console gives the  the number of traces of TPLs' user data access behavior in their code.
- The figure corresponds to Figure 3 in our paper.

### RQ_3.1 Reproduce results in RQ_3

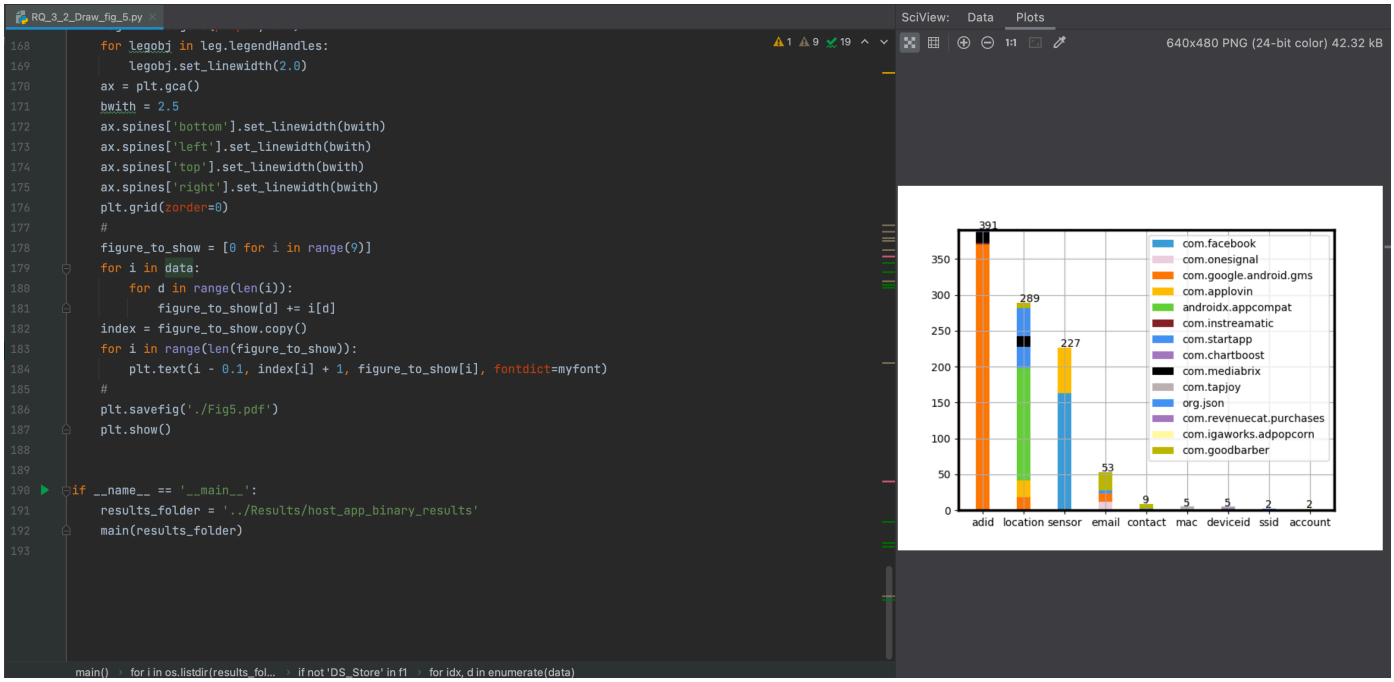
The users can run "*./Code/Part4_ResultsGenerator/RQ_3_1_Host_app_share_with_TPL.py*" to reproduce the results in RQ3. Before that, please replace the variable *results_path* in line 48 to the path you save the host apps' binary files' analysis results, which are generated by Step3.3, and replace the variable host_app_path in line_49 to the path you downloaded the dataset. The analysis results reproduce the results in Section IV.C.Results.Line.11 (ATPChecker identifies that  47.9% (220/459) host apps share PI with TPLs.).




- Please note that the reproduced results may differ from our results because of the difference of computer's hardware capacity. Some apps may fail due to time out settings in our code. We evaluate the performance of our tool on an iMac-2017 with Intel Core i7, 32 GB memory. If your computers' computation capacity is better than the given configuration, maybe mroe apps canbe analyzed; otherwise, more apps may failed.

### RQ_3.2 Reproduce Figure 5.

The users can run "*./Code/Part4_ResultsGenerator/RQ_3_2_Draw_fig_5.py*" to reproduce the Figure 5.  Before that, please replace the variable *results_folder* in line 191 to the the path you save the host apps' binary files' analysis results, which are generated by Step3.3.




- The figure corresponds to Figure 5 in our paper. The figure illustrate the number of traces that the apps shares

different kind of user data with TPLs.

- Please note that the reproduced results may differ from our results because of the difference of computer's hardware capacity. Some apps may fail due to time out settings in our code. We evaluate the performance of our tool on an iMac-2017 with Intel Core i7, 32 GB memory. If your computers' computation capacity is better than the given configuration, maybe mroe apps canbe analyzed; otherwise, more apps may failed.

**RQ_4_Identify_app_conpliance.py**

The users can run "*./Code/Part4_ResultsGenerator/RQ_4_Identify_app_conpliance.py*" to reproduce results in Table 5. Before that, please replace the variable *dataflow_folder* in line 102 to the the path you save the 256 host apps' binary files analysis results, which are generated by Step3.4 and replace the variable *pp_results_folder* in line 103 to the path you save the 256 host apps' privacy policy analysis results.

```
01 if __name__ == '__main__':
02     dataflow_folder = "../Results/256app_binary_results/"
03     pp_results_folder = "../Results/256app_pp_results"
04
05     #
06     TPLListFile = 'TPL_List_Results.txt'
07     TPLDataFile = 'TPL_Data_Results.txt'
08     TPLListWriter = open(TPLListFile, 'w')
09     TPLDataWrite = open(TPLDataFile, 'w')
10
```

The script will generate two results filese, namely 'TPL_List_Results.txt' and 'TPL_Data_Results.txt'. The 'TPL_List_Results.txt' gives the results that ATPChecker identifies whether the host app uses the TPL in their code and clearly discloses the usage of TPLs in their privacy policies. The content given in this file are as the following format:

```
app package name
  The conclusion whether the app claims the usage of TPL.
    The trace of TPL usage in app's binary code identified by ATPChecker.
    The sentences (if applicable) in app's privacy policy that claims the usage of
TPL.
```

Combing the host app's binary files analysis results and privacy policy analysis results, we manually calculate the metrics and provided in '*./Part4_ResultsGenerator/TPL_List_Metrics_ManualCheck.xlsx*'.

The 'TPL_Data_Results.txt' gives the results that ATPChecker identifies whether the host app shares data with TPLs and clearly disclose the data sharing behvaior in the privacy policies. The content given in the files are as the following format:

```
app package name
  The conclusion identified by ATPChecker that whether the app discloses the data
sharing behavior with TPL.
      The trace of data sharing behavior in app's binary code identified by
ATPChecker.
      The sentences (if applicable) in app's privacy policy that disclose the data
sharing behavior with TPL.
```

Combing the host app's binary files analysis results and privacy policy analysis results, we manually calculate the metrics and provided in '*./Part4_ResultsGenerator/TPL_Data_Metrice_ManualCheck.xlsx* '.