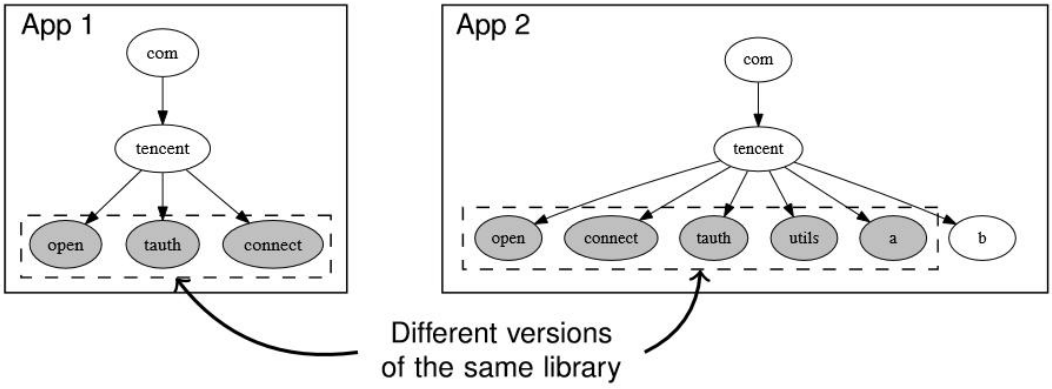
LibD 安卓第三方库探测：

1.优势：

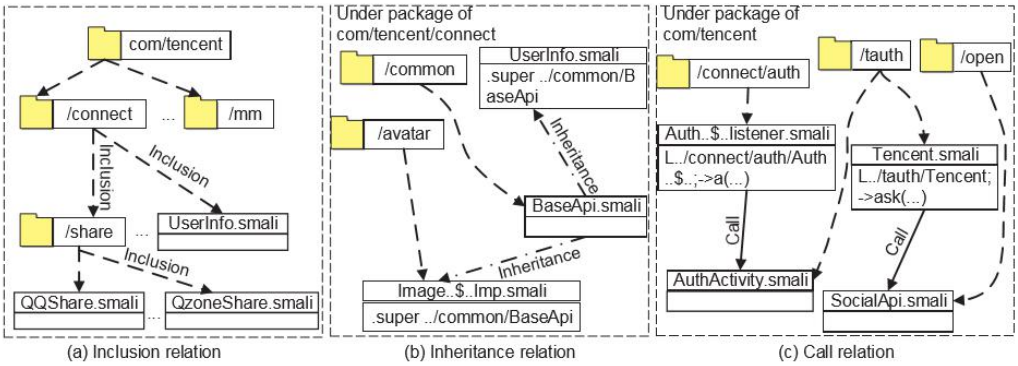
libD 工具克服了以往第三方库识别工具无法抵御包名混淆的缺陷，并能很好应对 app 版本更新带来的包结构变化。



2.流程概述

(1)反编译输入的 app，得到 IR(intermediate representation)。使用 APKtool 和 Androguard 实现。

(2)APKtool 得到 app 反汇编的树形结构，Androguard 得到方法，类，包之间的关系。这些关系包括：包含(inclusion)，继承(inheritance)，调用(call)。



(3)利用得到的信息建立库的实例：

A.建立同源包集合：同源包(Homogeny package)是指有继承和包含关系的高度相关的包。同源图(Homogeny graph):有向图 H = (V,E)，V 是 app 包的集合，E 是继承和包含关系的集合。同源包集合(Homogeny package union):同源图的弱连通子图

Algorithm 1: Homogeny package union construction

Input: Android app p
Output: Homogeny package union set $\mathcal{H}_p$

$\mathcal{H}_p \leftarrow \varnothing$; $\mathcal{H} \leftarrow \varnothing$,
$\mathcal{H}.V \leftarrow$ packages in the input app; /* V is the set of vertices. */
filter out packages in the root nodes in $\mathcal{H}$;
filter out Android official packages in $\mathcal{H}$;
$\mathcal{H}.E \leftarrow$ inclusion relation set; /* E is the set of edges. */
$\mathcal{H}.E \leftarrow \mathcal{H}.E \cap$ inheritance relation set;
for *each weakly connected component g in* $\mathcal{H}$ **do**
    $\mathcal{H}_p$.add(g);
return $\mathcal{H}_p$

B.首先建立调用图(call graph)(在同源包集合间的调用，然后过滤掉 application call 以及 ghost call 最后在弱连通子图中找到根节点，从根结点能到达的同源包集合作为库的实例输出。(若找不到根结点，则直接输出该同源包集合作为一个库的实例)

Algorithm 2: Library instance construction

Input: Homogeny package union set $\mathcal{H}_p$
Output: Library instance set $\mathcal{I}_l$

```
1  I ← ∅;
2  I.V ← H_p; /* V is the set of vertices. */
3  for any union u_1 and u_2 in I do
4      if there is a call relation in ⟨u_1, u_2⟩ then
5          add ⟨u_1, u_2⟩ in I.E; /* E is the set of edges. */
6  filter out application code-related calls in I;
7  filter out ghost calls in I;
8  for each weakly connected component g in I do
9      if there are root nodes in g then
10         for each root do
11             cl ← reachable components from this root;
12             I_l.add(cl);
13     else
14         I_l.add(g);
15 return I_l
```

(4)生成特征
一个库的实例包含着一个或多个同源包集合。特征提取从方法(method)级别开始，首先建立方法的控制流图(CFG,control flow graph)。程序流图每一部分的特征是有该部分 opcode 的 hash 值。将得到 hash 值按深度优先排序。然后将类中所用的方法按方法 hash 值排序，并再次进行 hash 得到类的特征值，最后按照同样的方法得到库的实例的特征值
(5)第三方库检测
利用前面提到的库的实例建立和特征值计算方法，在大量的 app 中，按照规定的阈值，对实例进行聚类，并判定为第三方库(判定出的第三方库列表在 liblist 文件夹中)。最后当使用 libD 对输入的 apk 进行分析时只需要将输入 apk 种库实例的特征值与 liblist 文件夹中的第三方库列表进行比对。

3.使用方法
环境：Linux 操作系统
预安装：(1) Python 2.7
(2) OpenJDK 1.7.0 or later
(3) Apktool 2.0.0 or later. http://ibotpeaches.github.io/Apktool/
(4) Androguard 1.9. https://github.com/androguard/androguard
使用：
python libd_v_0.0.1.py 目标 apk 路径 输出文件夹路径 阈值设定(liblist 文件夹中的 csv 文件)

4.链接
https://github.com/IIE-LibD/libd