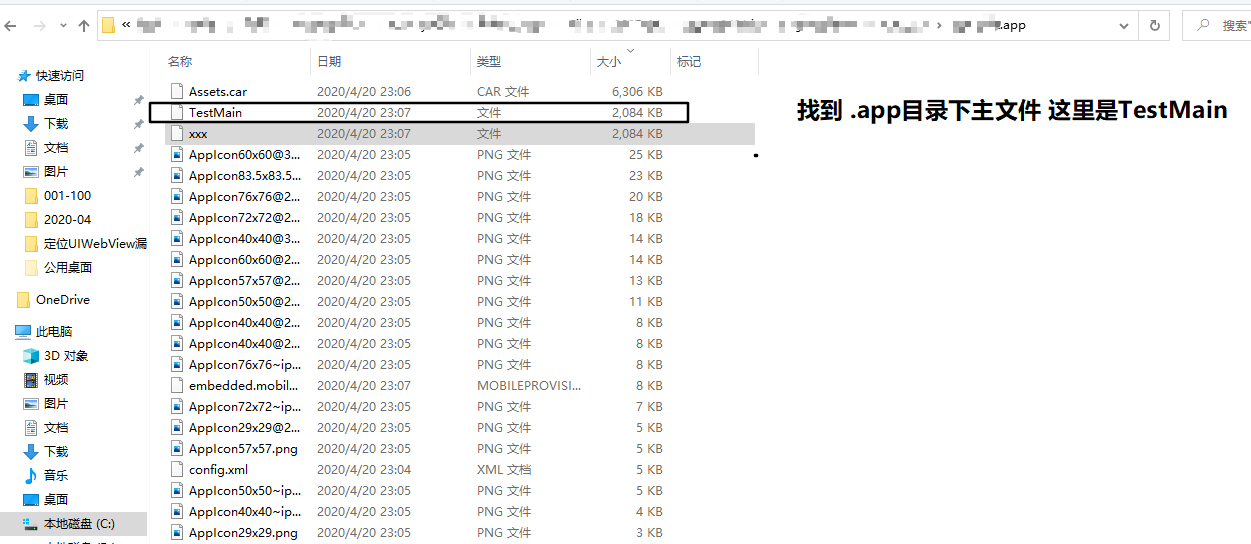
**c库api相关**

**首先解压ipa 找到主文件**



**找到报告里的风险项**



**可以看到这里有3个不安全的 api**

memcpy

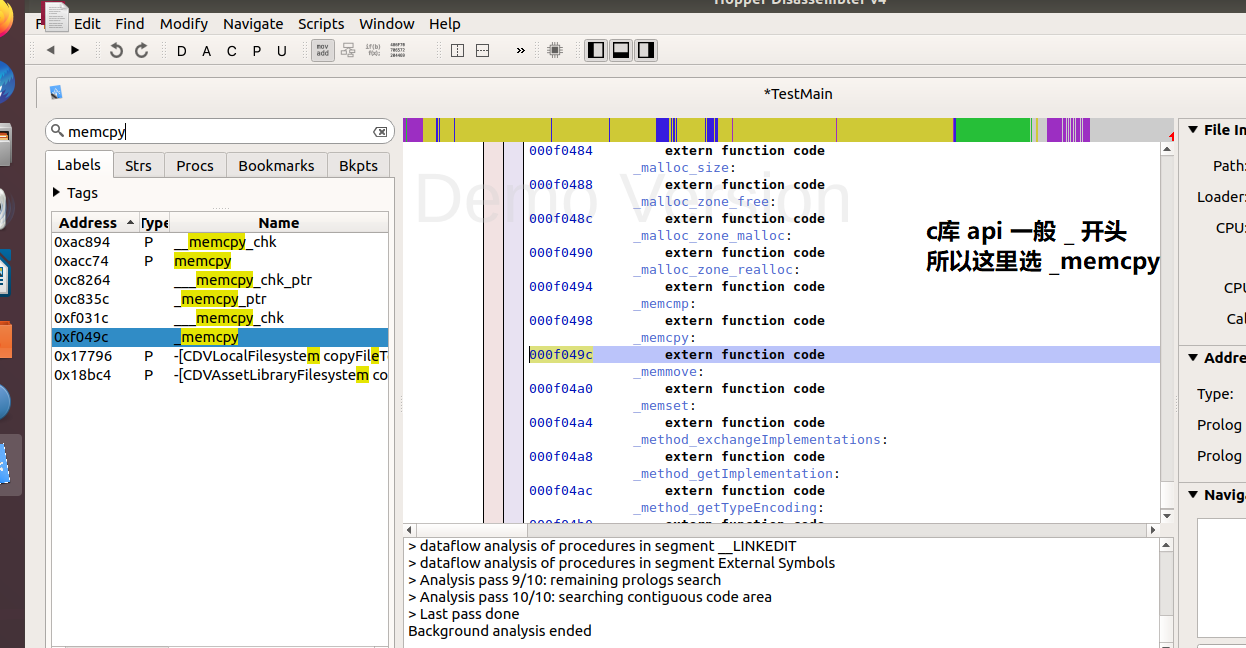
sscanf

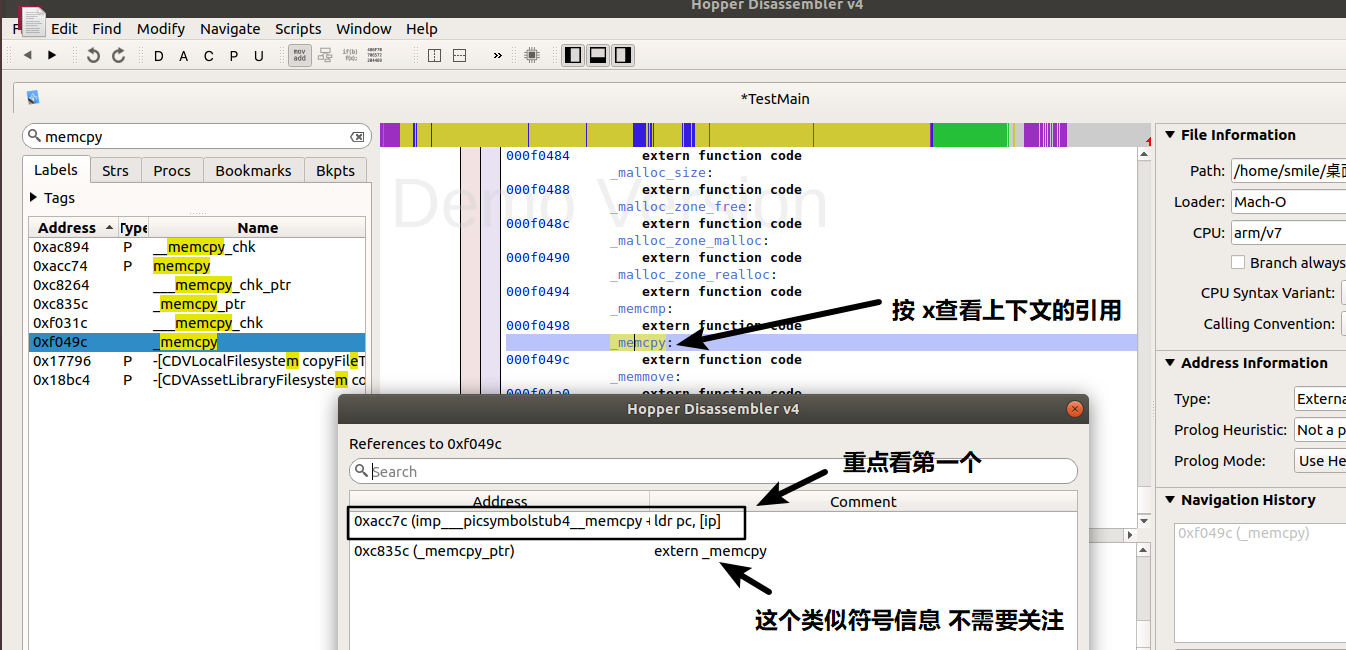
strlen

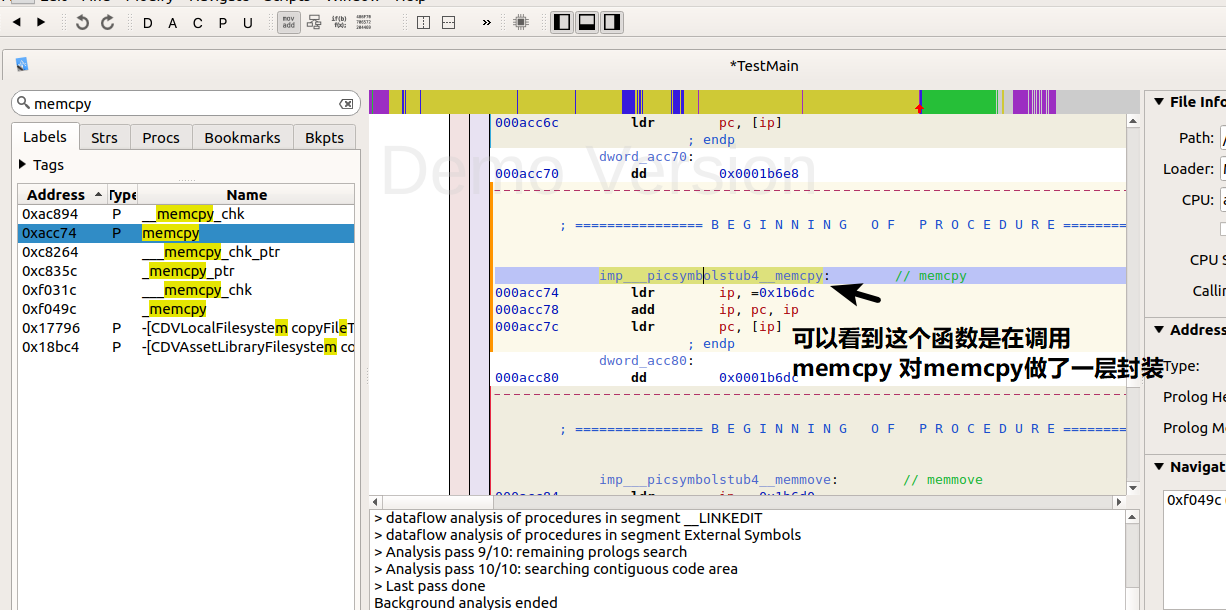
**打开 hopper**



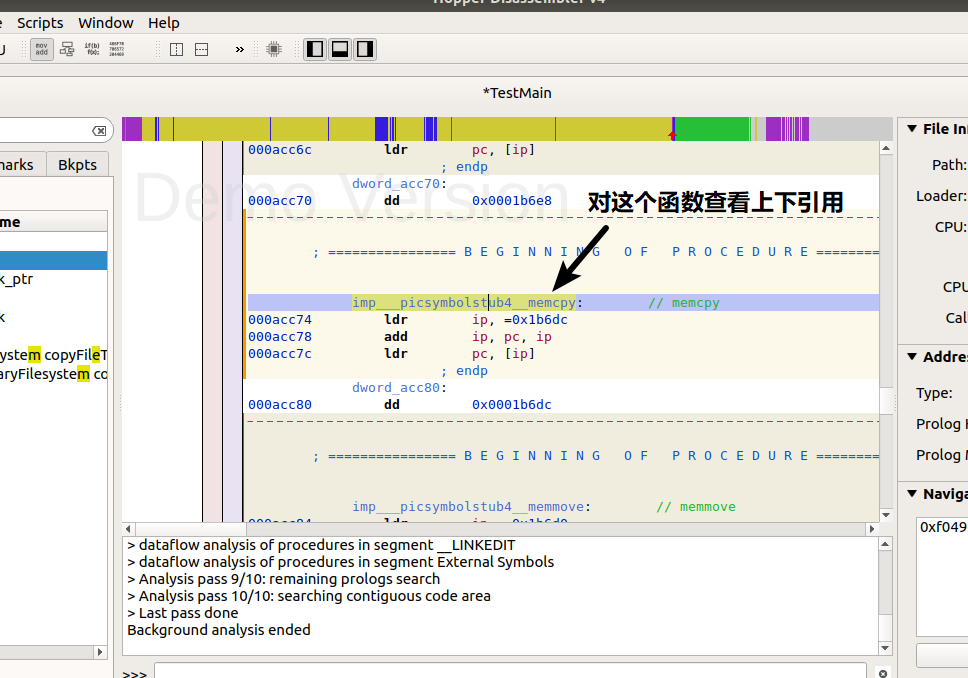
这个工具是个跨平台的工具   linux 和 mac平台打开都可以使用

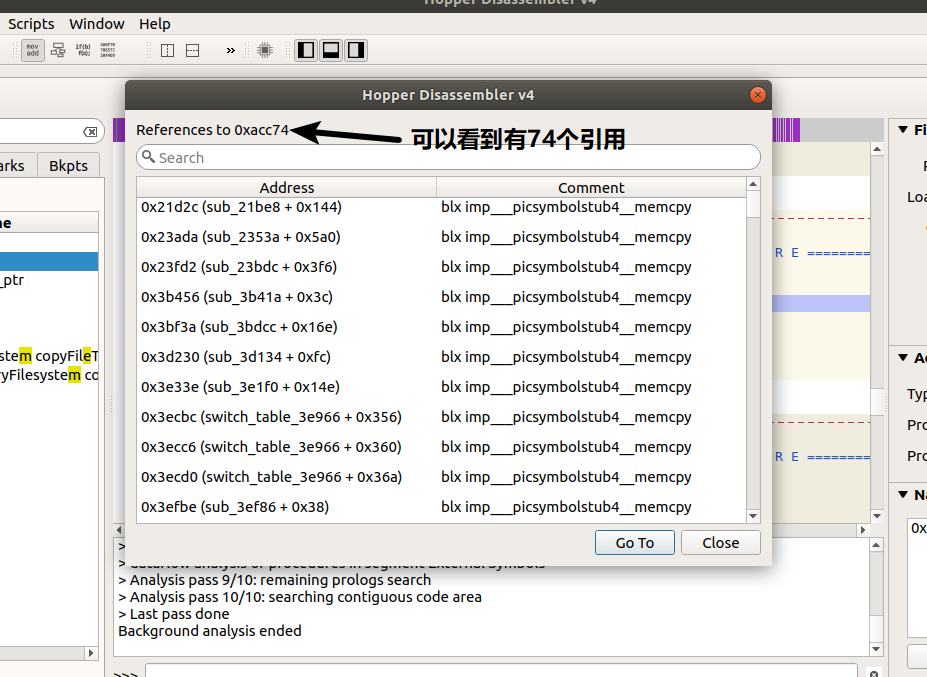


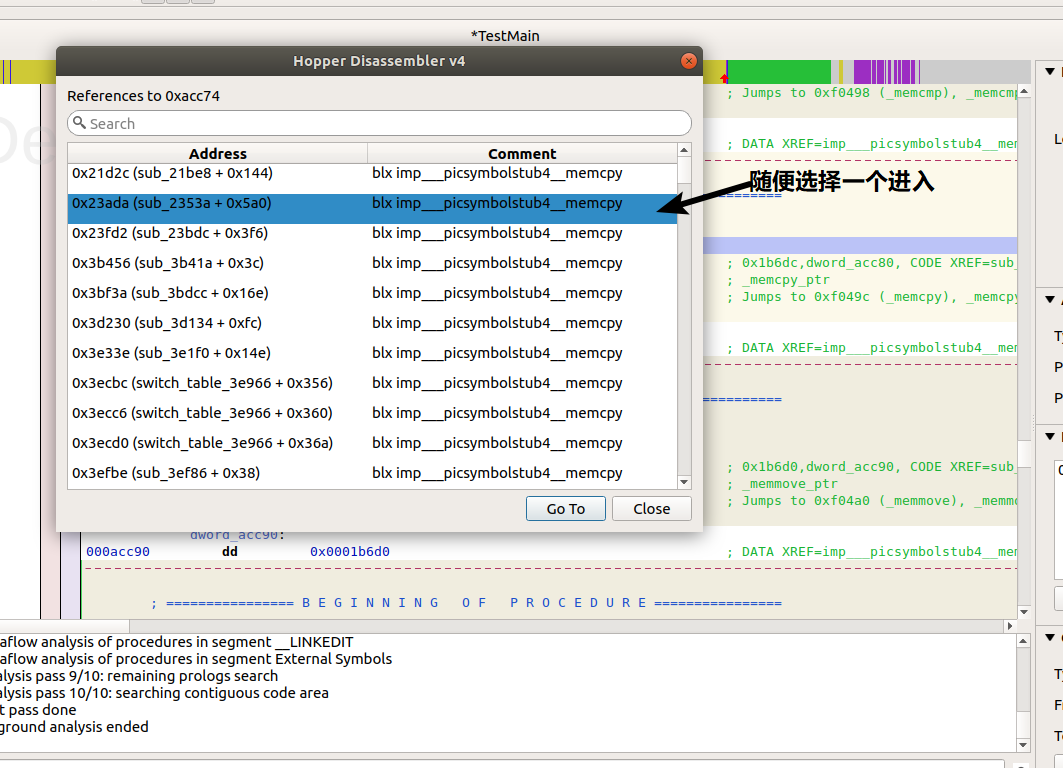


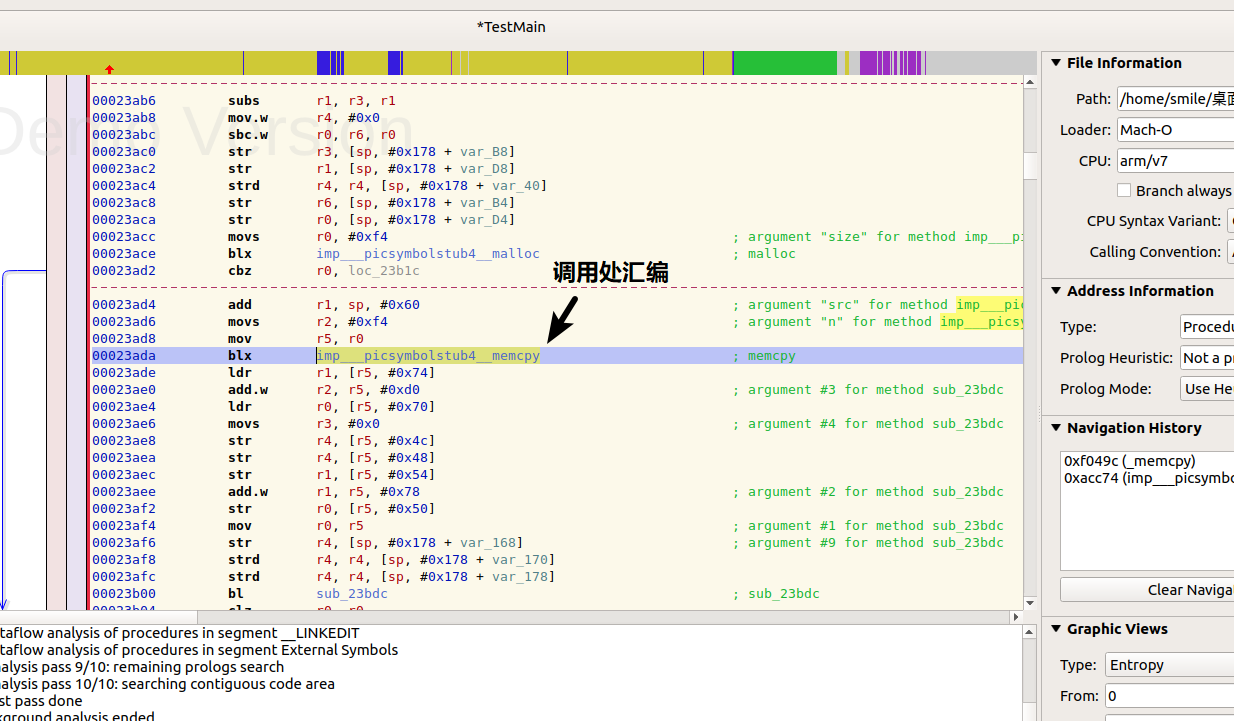


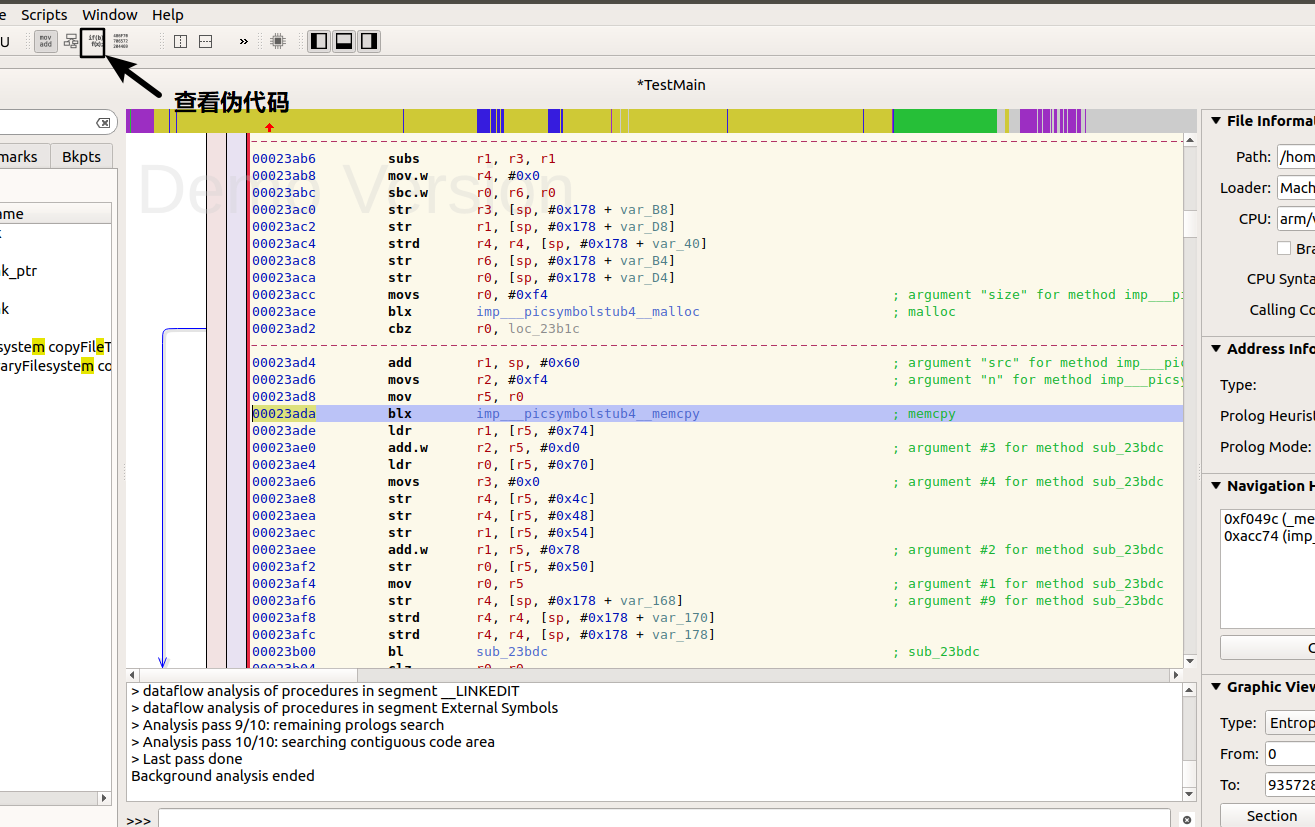
这里如果没有进行封装 可以直接看上下文引用

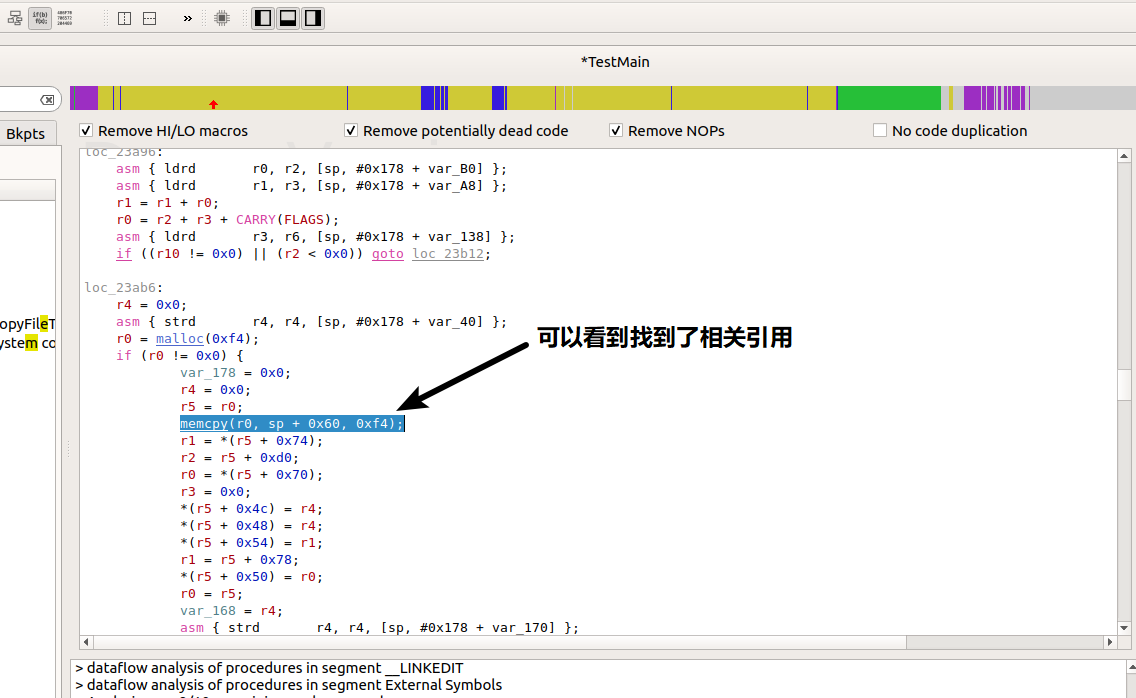












sscanf

strlen

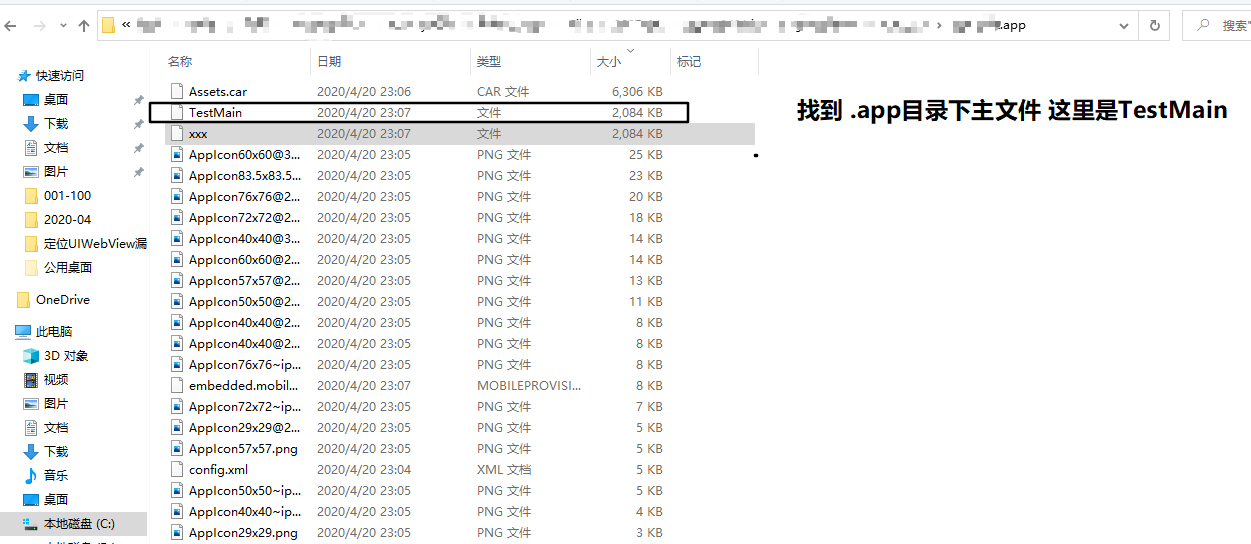
 同上

后续可以根据 上层函数引用  伪代码逻辑 和字符串特征等 确定风险 api 是在 静态的编译的第三方sdk 还是主文件

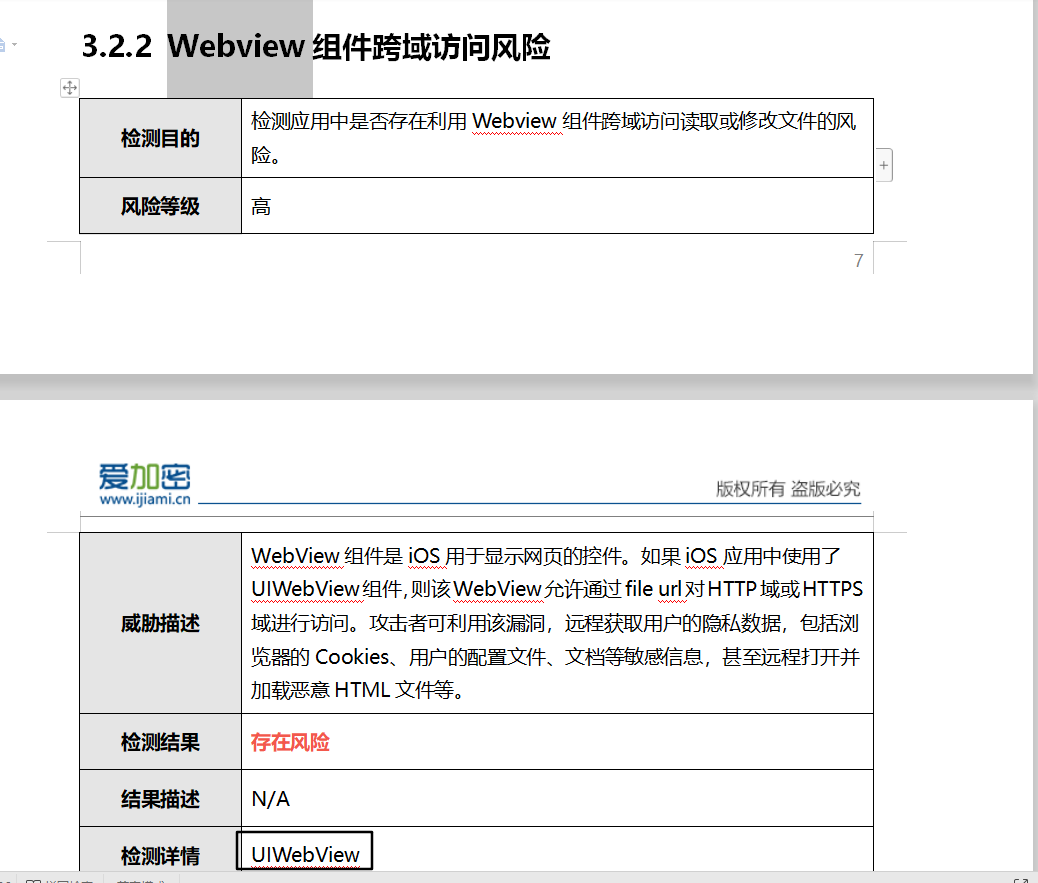
定位修改

**UIWebView 相关**

**首先解压ipa 找到主文件**



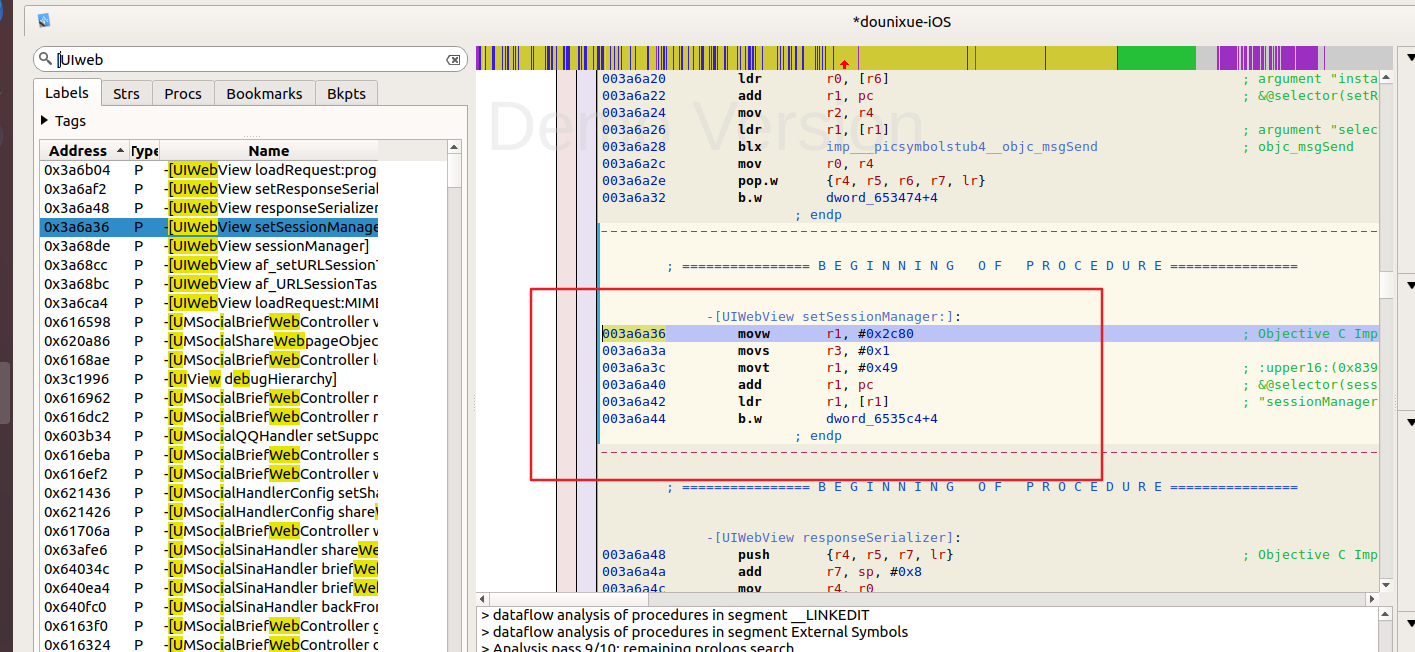
**找到报告里的风险项**



**打开 hopper**



这个工具是个跨平台的工具   linux 和 mac平台打开都可以使用



输入 UIWebView 可以看到 已经看到了很多引用方法 此刻函数头按 x找到相关上下文引用。

后续可以根据 上层函数引用  伪代码逻辑 和字符串特征等 确定风险 api 是在 静态的编译的第三方sdk 还是主文件

然后定位修改