Android应用逻辑漏洞半自动化挖掘思路

mp.weixin.qq.com/s/tFFe_LOs0e1Po8nj9ifmKg

大清早起来就看到F-Secure LABS团队（以前叫MWR，就是那支用13个逻辑漏洞攻破Chrome浏览器的团队，是Pwn2Own专业户）发了一篇文章“Automating Pwn2Own with Jandroid”(https://labs.f-secure.com/blog/automating-pwn2own-with-jandroid/)，讲述如何利用Jandroid实现Android应用逻辑漏洞的半自动化挖掘思路。

专注逻辑漏洞有一些好处，尤其是作比赛用途的，撞洞率较低，且利用稳定，一般都不用搞什么内存布局控制的。

MWR尤其擅长此类漏洞的挖掘，之前就在Mobile Pwn2Own上攻击破过华为手机、三星手机和Chrome浏览器。

文中介绍了Jandroid (https://github.com/FSecureLABS/Jandroid)这款开源工具，该工具要求python 3.4以上版本才能运行，支持 `apk` 、 `dex` 、 `system.img` 、 `ext4` 文件解析。

```
python3 src/jandroid.py -h

---------------------------
           JANDROID
---------------------------

usage: jandroid.py [-h] [-f FOLDER] [-p [{android}]] [-e [{device,ext4,img}]]
                   [-g [{neo4j,visjs,both}]]

A tool for performing pattern matching against applications.

optional arguments:
  -h, --help            show this help message and exit
  -f FOLDER, --folder FOLDER
                        app分析目录，所以支持应用的批量分析
  -p [{android}], --platform [{android}]
                        支持的平台，目前仅支持android平台
  -e [{device,ext4,img}], --extract [{device,ext4,img}]
                        支持从连接设备、ext4、system.img中提取应用
  -g [{neo4j,visjs,both}], --graph [{neo4j,visjs,both}]
                        支持检测结果的图表显示
```

它通过定义json模板来标记污点传播路径，比如拥有 `android.intent.category.BROWSABLE` 浏览器打开权限的Activity，再查找 `Landroid/webkit/WebView;->addJavascriptInterface` 看是否存在JavaScript接口，以判断是否可能存在远程攻击的条件，但这种只能是半自动化辅助，还需要人工进一步确认。

模板示例：

```
{
    "METADATA": {
        "NAME": "JSbridgeBrowsable"
    },
    "MANIFESTPARAMS": {
        "BASEPATH": "manifest->application->activity OR manifest->application->activity-
alias",
        "SEARCHPATH": {
            "intent-filter": {
                "action": {
                    "LOOKFOR": {
                        "TAGVALUEMATCH": "<NAMESPACE>:name=android.intent.action.VIEW"
                    }
                },
                "category": {
                    "LOOKFOR": {
                        "TAGVALUEMATCH": "
<NAMESPACE>:name=android.intent.category.BROWSABLE"
                    }
                },
                "data": {
                    "RETURN": ["<NAMESPACE>:host AS @host", "<NAMESPACE>:scheme AS
@scheme"]
                }
            }
        },
        "RETURN": ["<smali>:<NAMESPACE>:name AS @activity_name"]
    },
    "CODEPARAMS": {
        "SEARCH": {
            "SEARCHFORCALLTOMETHOD": {
                "METHOD": "Landroid/webkit/WebView;->addJavascriptInterface",
                "RETURN": "<class> AS @web_view"
            }
        },
        "TRACE": {
            "TRACEFROM": "<method>:@web_view[]->loadUrl(Ljava/lang/String;)V",
            "TRACETO": "<class>:@activity_name",
            "TRACELENGTHMAX": 10,
            "RETURN": "<tracepath> AS @tracepath_browsablejsbridge"
        }
    },
    "GRAPH": "@tracepath_browsablejsbridge WITH <method>:<desc>:<class> AS
attribute=nodename"
}
```

各字段含义看示例就好了，这里不作详解。读者也可参考F-Secure发的文章，里面有详解。

总结起来，模板支持：

1. AndroidManifest.xml的匹配搜索

2. smali代码的匹配搜索

3. 传播路径的图表显示，以及显示的文件格式定义

4. 函数调用参数追踪

5. 函数调用的起点与终点定义、追踪以及追踪深度

我直接找了个apk分析，一运行就出现以下错误：

```
python3 src/jandroid.py -f ./apps -g visjs
```

```
Traceback (most recent call last):

  File "src/jandroid.py", line 408, in <module>
    inst_jandroid.fn_main()
  File "src/jandroid.py", line 227, in fn_main
    self.pull_source
  File "/Volumes/Macintosh/Users/riusksk/Android-Security/工
具/Jandroid/src/plugins/android/main.py", line 51, in fn_start_plugin_analysis
    app_pull_src
  File "/Volumes/Macintosh/Users/riusksk/Android-Security/工
具/Jandroid/src/plugins/android/requirements_checker.py", line 53, in
fn_perform_initial_checks
    raise JandroidException(
NameError: name 'JandroidException' is not defined
```

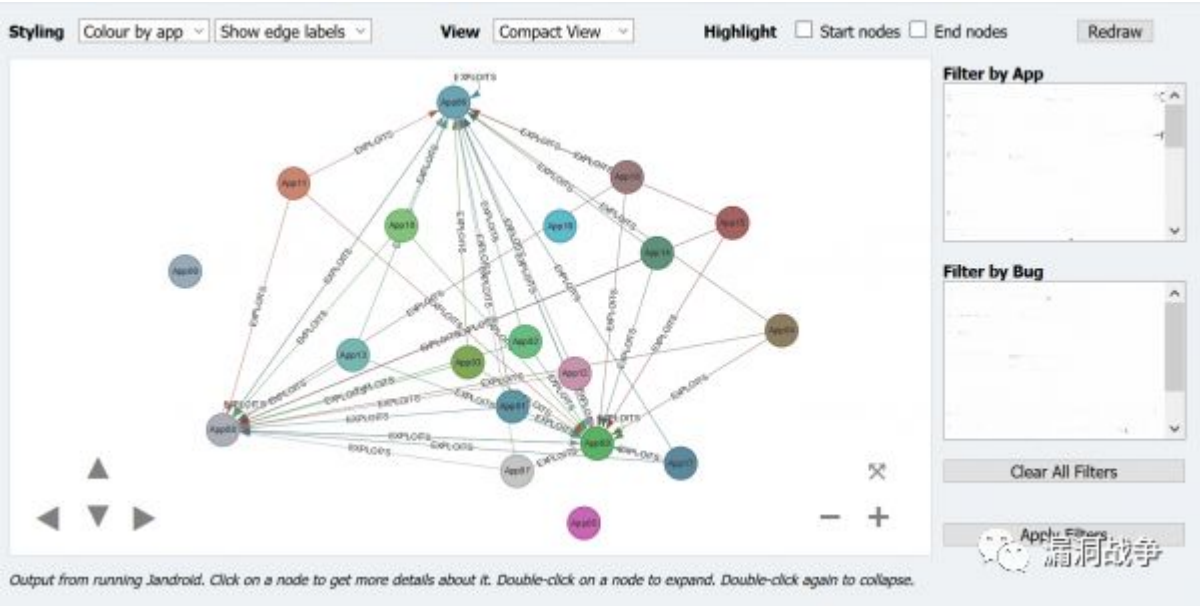
直接在 `Jandroid/src/plugins/android/requirements_checker.py` 开头加以下代码即可解决：

```
from common import JandroidException
```

运行效果：

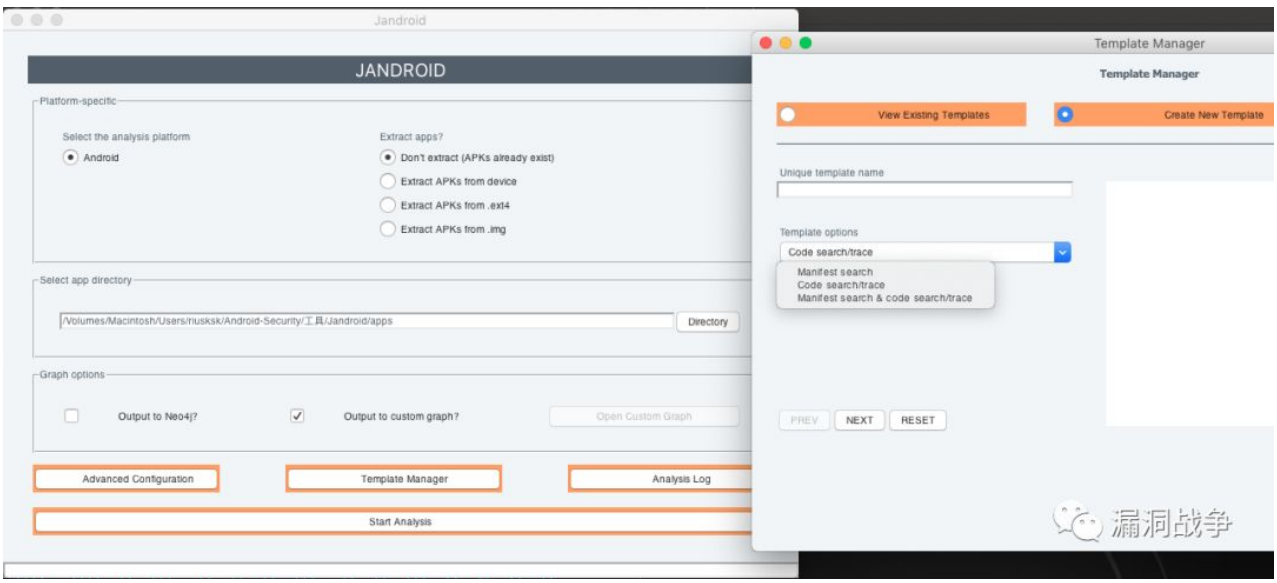
```
python3 src/jandroid.py -f ./apps -g visjs

----------------------------
           JANDROID
----------------------------

INFO     Creating template object.
INFO     1 potential template(s) found.
DEBUG    Parsing /Volumes/Macintosh/Users/riusksk/Android-Security/工
具/Jandroid/templates/android/sample_basic_browsable_jsbridge.template
INFO     Initiating Android analysis.
INFO     Performing basic checks. Please wait.
INFO     Basic checks complete.
INFO     Beginning analysis...
DEBUG    1 app(s) to analyse, using 2 thread(s).
DEBUG    Created worker process 0
DEBUG    Created worker process 1
DEBUG    AnalyzeAPK
DEBUG    Analysing without session
INFO     Analysing ctrip.android.view_8.13.0_1248.apk in worker thread 0.
DEBUG    AXML contains a RESOURCE MAP
DEBUG    Start of Namespace mapping: prefix 47: 'android' --> uri 48:
'http://schemas.android.com/apk/res/android'
DEBUG    START_TAG: manifest (line=2)
DEBUG    found an attribute:
{http://schemas.android.com/apk/res/android}versionCode='b'1248''
DEBUG    found an attribute:
{http://schemas.android.com/apk/res/android}versionName='b'8.13.0''
DEBUG    found an attribute:
......
DEBUG    Settings basic blocks childs
DEBUG    Creating exceptions
DEBUG    Parsing instructions
DEBUG    Parsing exceptions
DEBUG    Creating basic blocks in Landroid/support/constraint/solver/LinearSystem;-
>createRowDimensionPercent(Landroid/support/constraint/solver/LinearSystem;
Landroid/support/constraint/solver/SolverVariable;
Landroid/support/constraint/solver/SolverVariable;
Landroid/support/constraint/solver/SolverVariable; F
Z)Landroid/support/constraint/solver/ArrayRow; [access_flags=public static] @ 0x199210
......
DEBUG    Looking for subclasses of Lctrip/business/map/SimpleOverseaMapActivity;
DEBUG    ctrip.android.view_8.13.0_1248.apk took 349 seconds to analyse.
DEBUG    Finished analysing ctrip.android.view_8.13.0_1248.apk with output {'bug_obj':
{'JSbridgeBrowsable': False}, 'graph_list': []}.
INFO     Finished analysing apps.
INFO     Creating custom graph.
INFO     Custom graph can be found at /Volumes/Macintosh/Users/riusksk/Android-Security/
工具/Jandroid/output/graph/jandroid.html
INFO     All done.
```

输出结果会在上面jandroid.html中显示，但由于我这里没有检测到满足JSbridgeBrowsable条件的代码，因此html里面的图是空的。如果有满足条件的代码，会得到类似如下的图：



Jandroid还提供有GUI操作界面，包括模板创建功能，所以使用也很方便，运行以下命令即可打开：

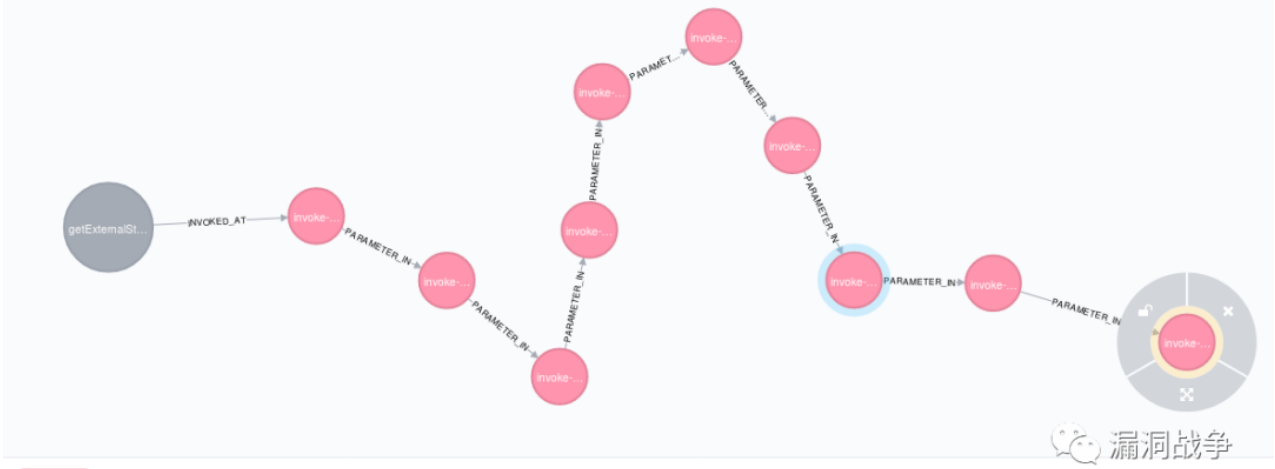
```
python3 gui/jandroid_gui.py
```



比如追踪DexClassLoader.loadClass加载外部dex文件的情况：

再举个实例，下图是MWR当初分析三星时，一个Unzip目录穿越漏洞的函数传播路径图，漏洞被用于Mobile Pwn2Own 2017：

所以，Jandroid还是非常适合用来挖掘逻辑漏洞的辅助工具，核心思想依然是污点追踪的思路，操作简单，可视化效果也很好。基于模板的定制化，增加了其运用的灵活性，尤其对于复杂的业务逻辑设计，很适合作定制化地批量检测，但依然需要人工分析确认，并非完全自动化的。



Invocation **<id>:** 316678 **code:** invoke-virtual {v0, v10}, Ldalvik/system/DexClassLoader;->loadClass(Ljava/lang/String;)Ljava/lang/Class; **line_no:** 218