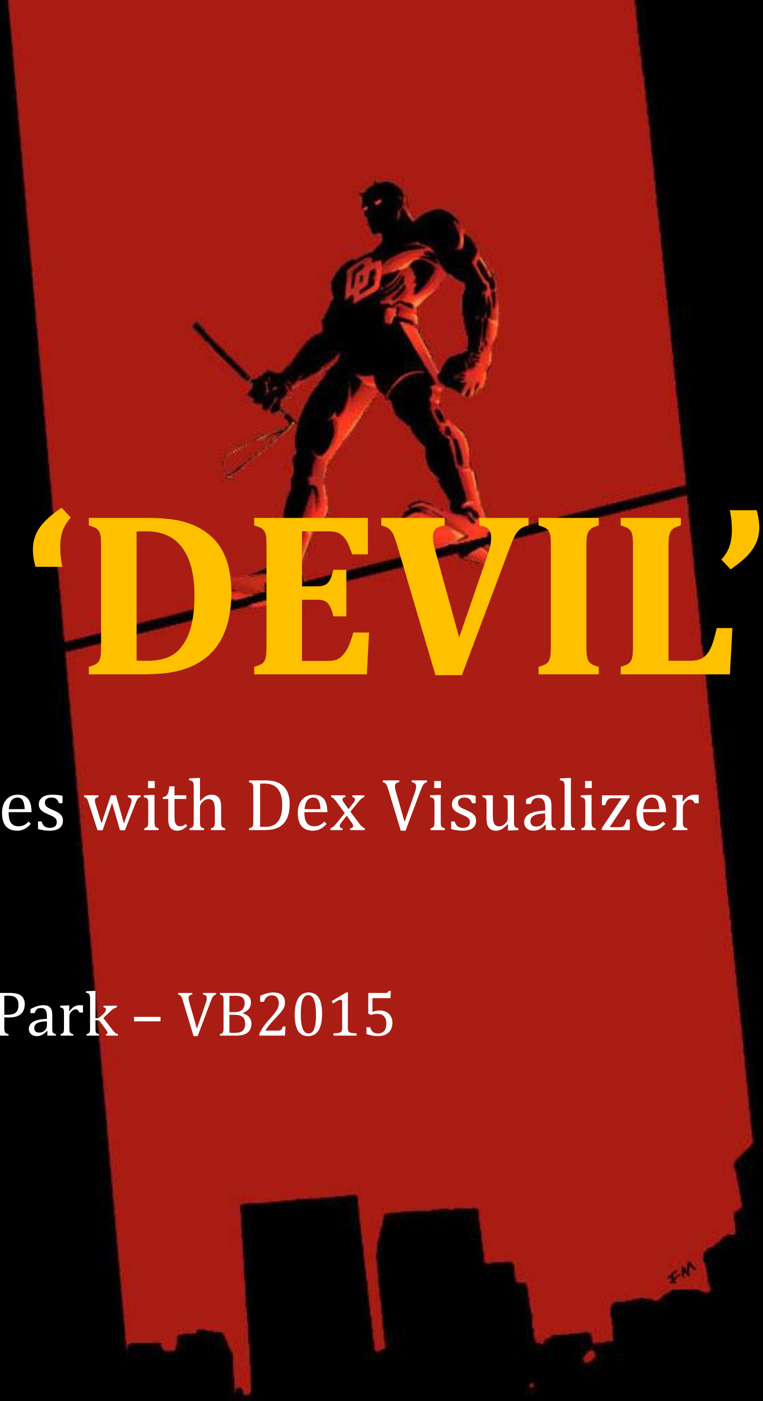


DARE ‘DEVIL’

beyond your senses with Dex Visualizer

Jun Yong Park – VB2015

whoami

Senior Principal Researcher / Architect

jypark@ahnlab.com

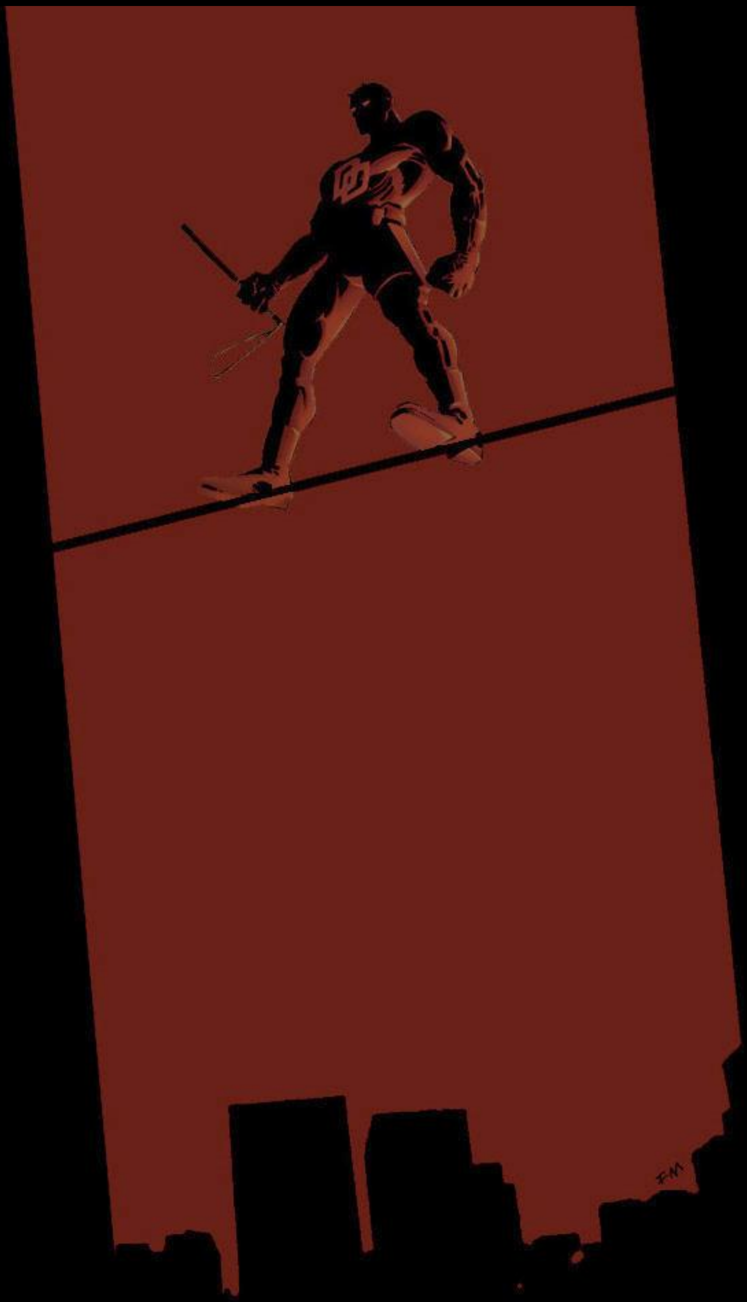
Security researcher @AhnLab since 2004

in recent years enjoy reversing and visualizing

Android malware

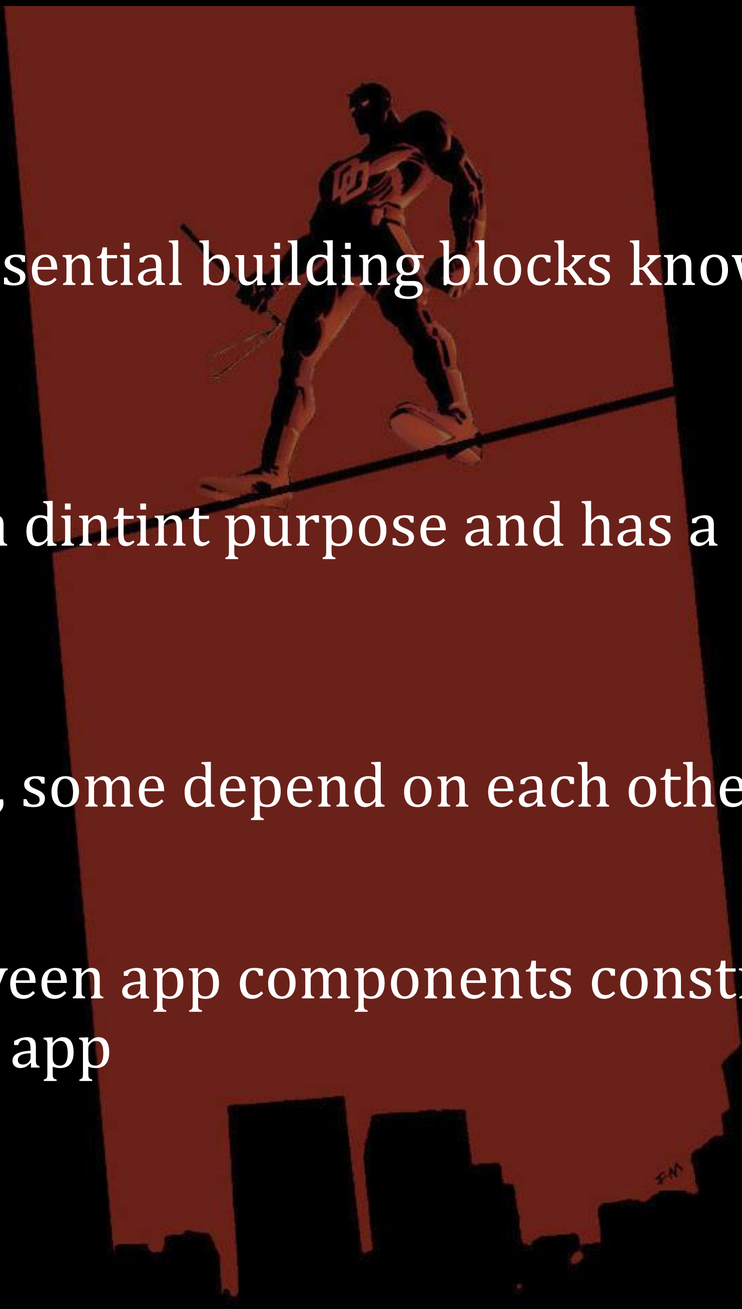
Agenda

- Motivations
- App lifecycle [graph]
- DEVIL
- How-to
- App lifecycling
- Case studies
- Conclusion



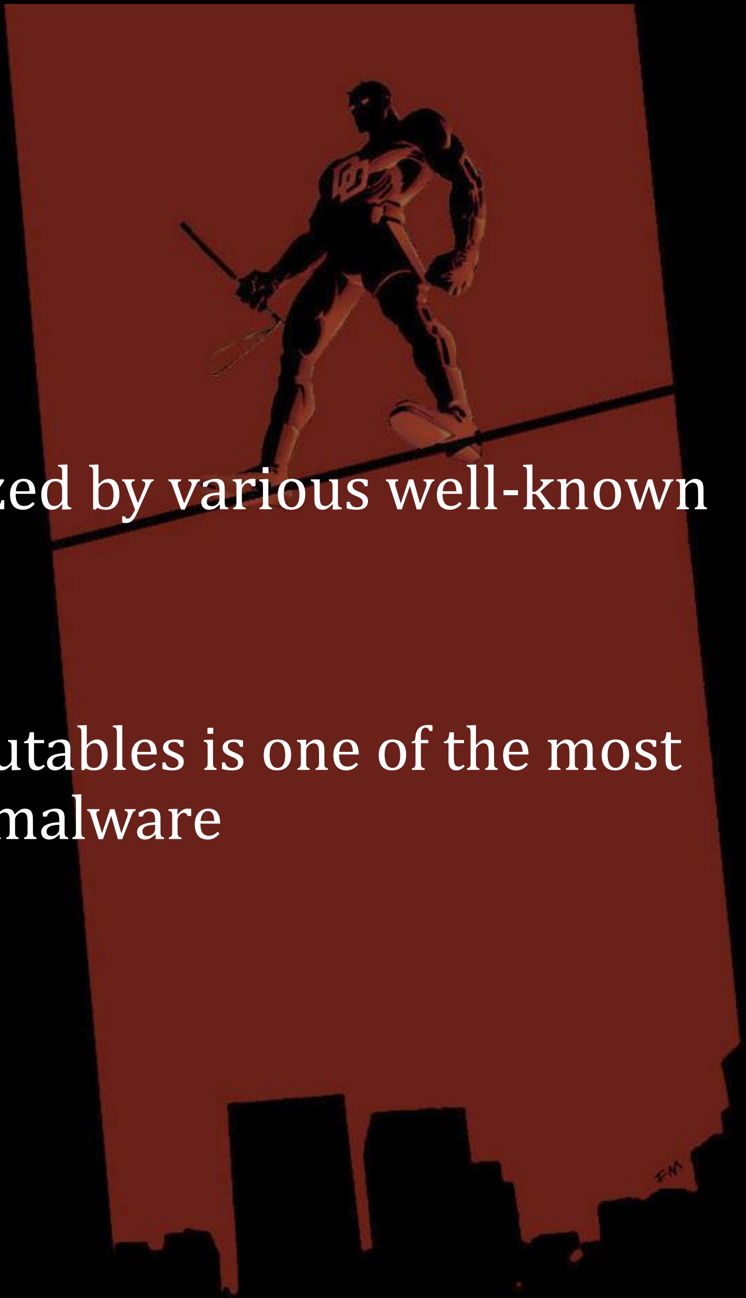
See the wood for the trees

App Lifecycle



- Every *Android* app has essential building blocks known as app components
- Each component serves a dintint purpose and has a distinct lifecycle
- Some interact each other, some depend on each other
- These relationships between app components construct the lifecycle of an *Android* app

App Lifecycle Graph

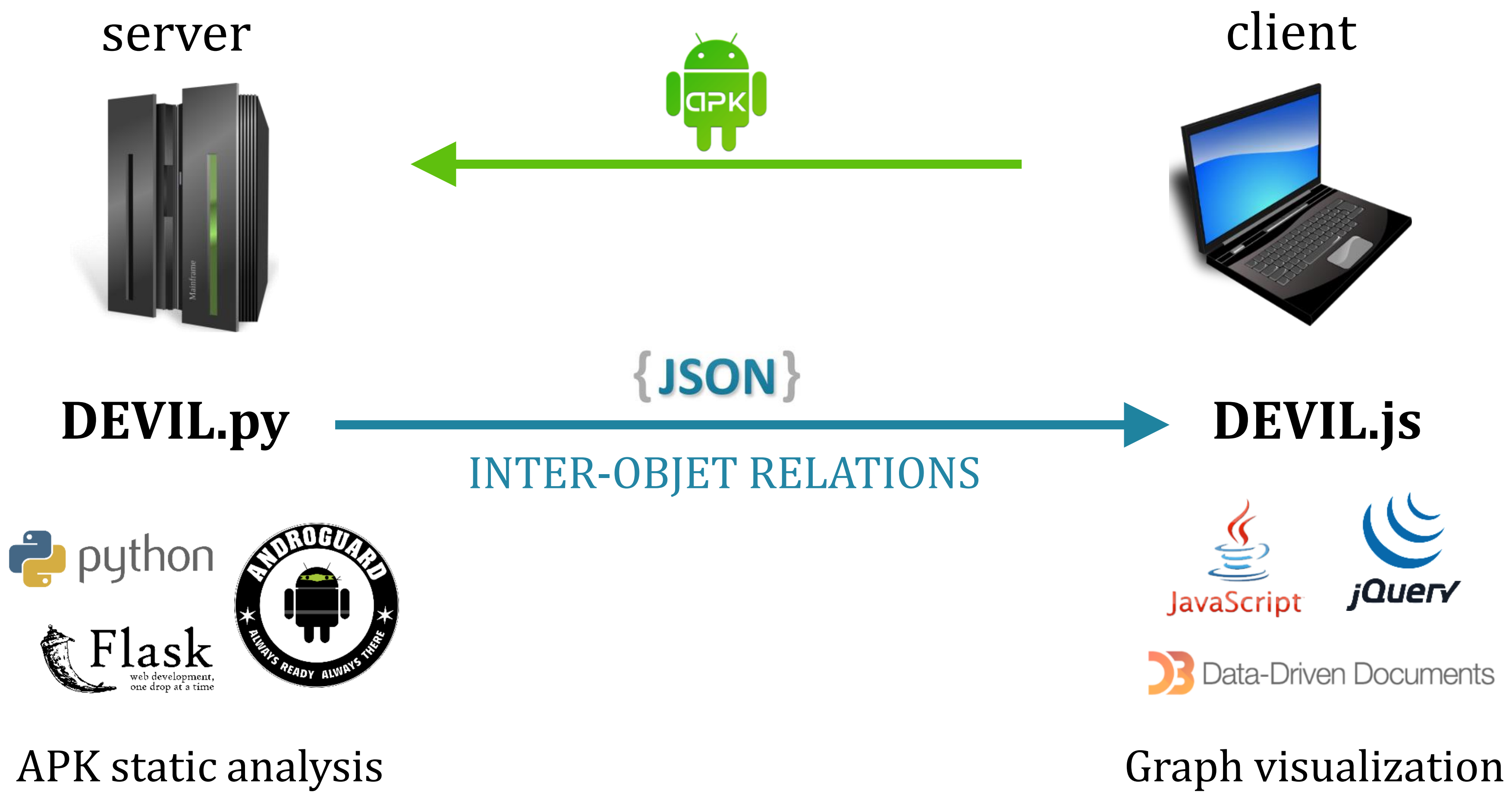


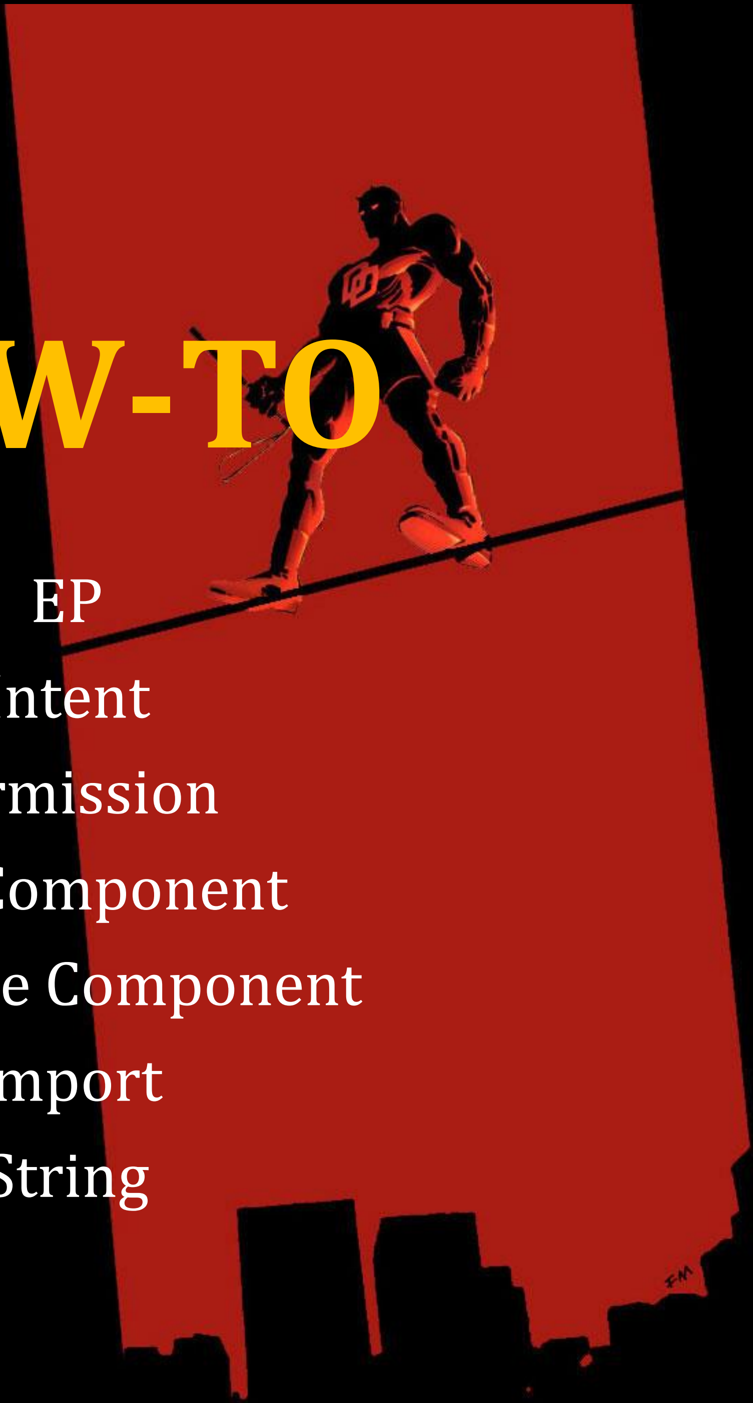
- A lifecycle can be visualized by various well-known graph algorithm
- The visualization of executables is one of the most effective ways to identify malware

Dare ‘DEVIL’

Dex Visualizer

DEVIL


server
client
APK
{JSON}
DEVIL.py
DEVIL.js
INTER-OBJET RELATIONS
python
Flask
web development, one drop at a time
ANDROGUARD
ALWAYS READY ALWAYS THERE
JavaScript
jQuery
Data-Driven Documents
APK static analysis
Graph visualization



HOW-TO

EP

Intent

Permission

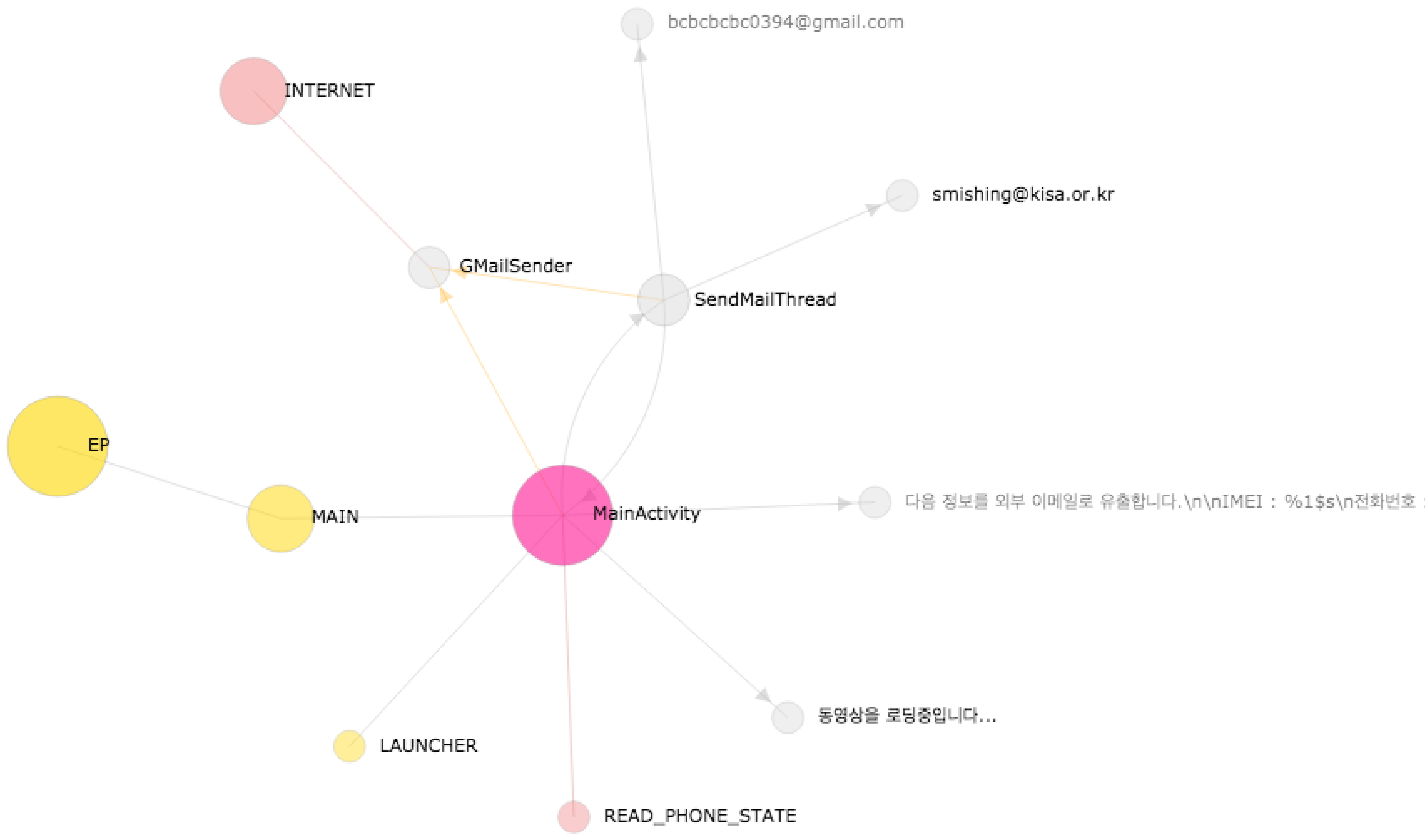
App Component

Runnable Component

Import

String

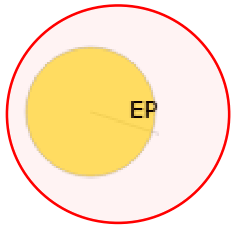
Test/PNStealer



Entry Point (1/7)

Android-Test/PNStealer

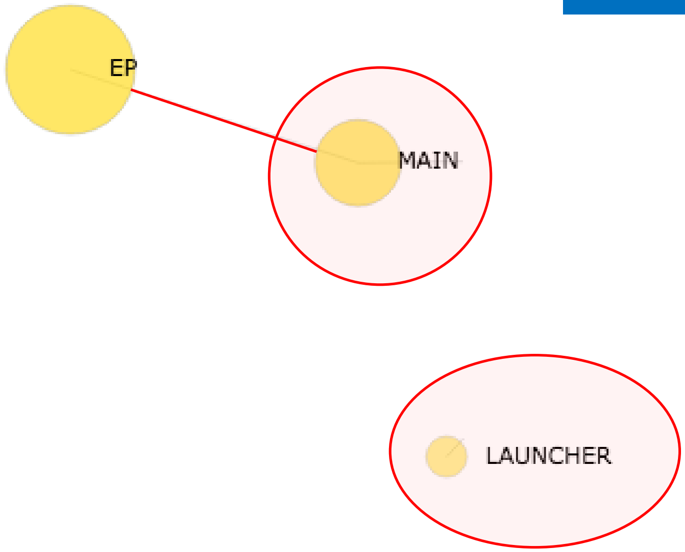
- First, only one abstract node, EP

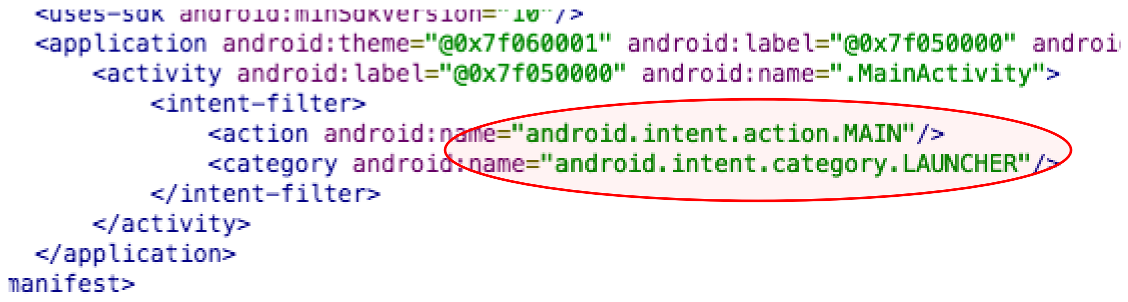


Intent (2/7)

Android-Test/PNStealer

- ❖ reading AndroidManifest.xml
- ❖ emulating bytecodes
- ❖ tracing the life of objects

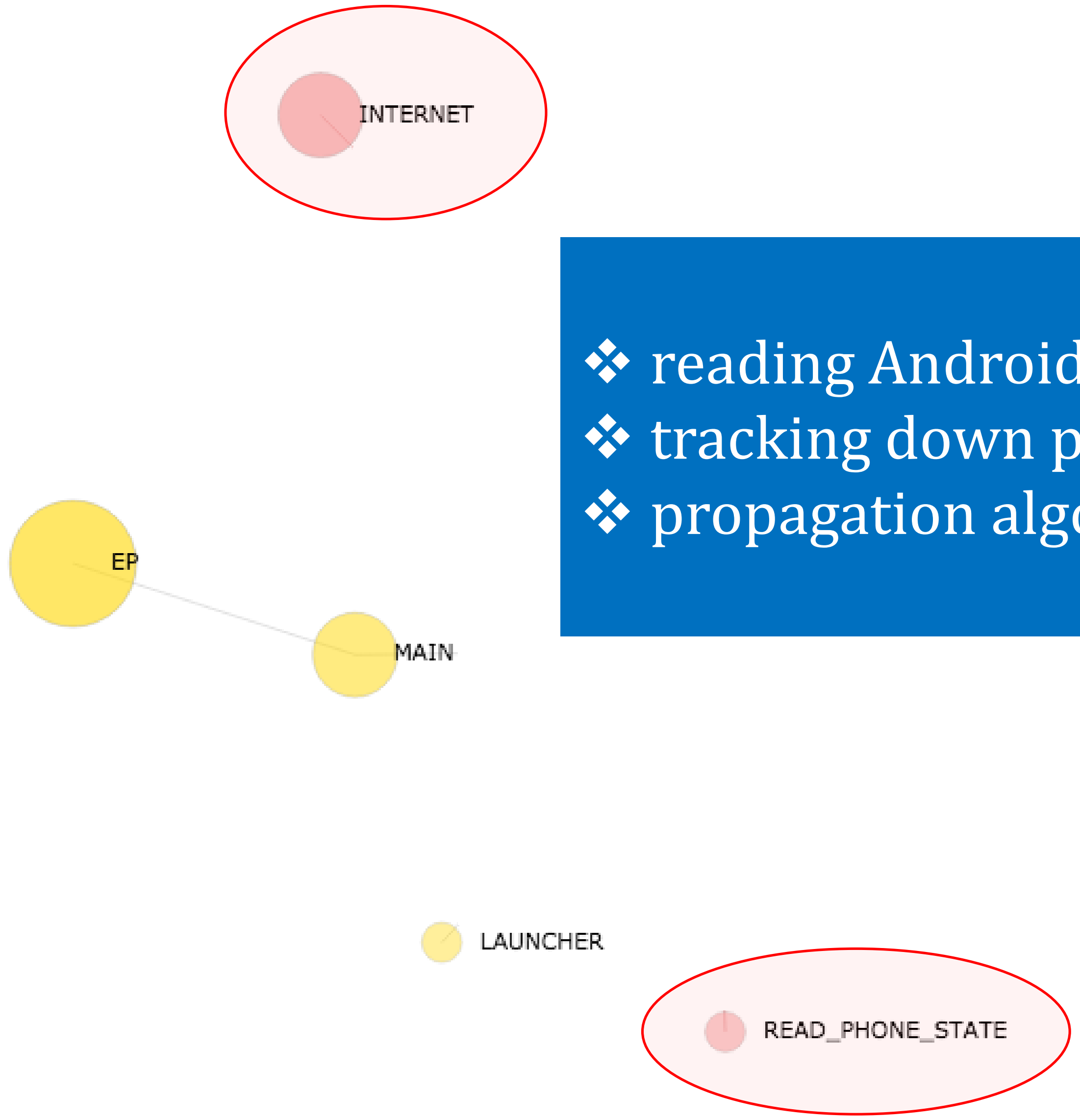




```
  <uses-sdk android:minSdkVersion="10"/>
  <application android:theme="@0x7f060001" android:label="@0x7f050000" androi
      <activity android:label="@0x7f050000" android:name=".MainActivity">
          <intent-filter>
              <action android:name="android.intent.action.MAIN"/>
              <category android:name="android.intent.category.LAUNCHER"/>
          </intent-filter>
      </activity>
  </application>
manifest>
```

Permission (3/7)

Android-Test/PNStealer

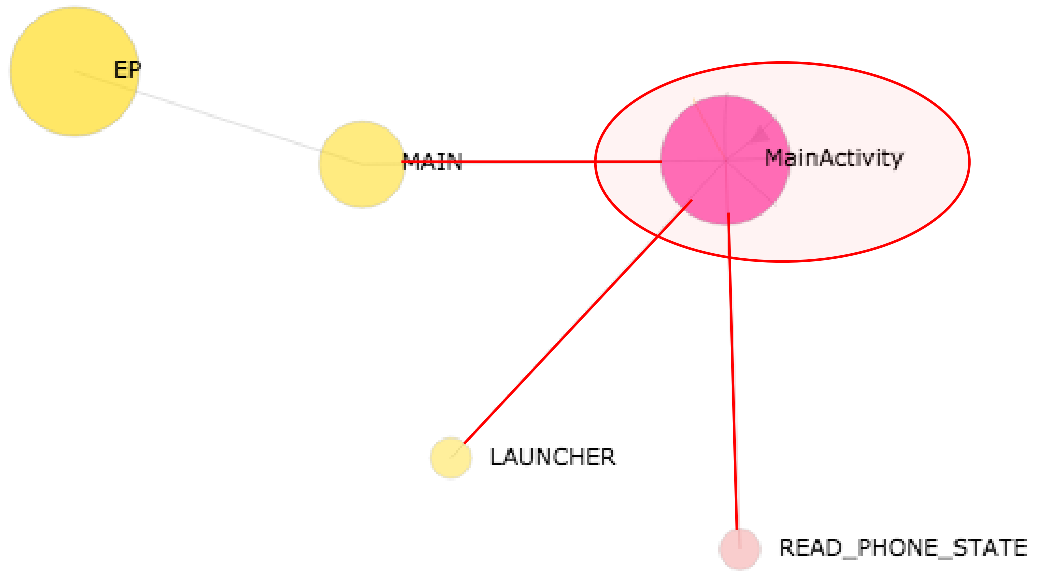


- reading AndroidManifest.xml
- tracking down permission usages
- propagation algorithm

App Component (4/7)

Android-Test/PNStealer

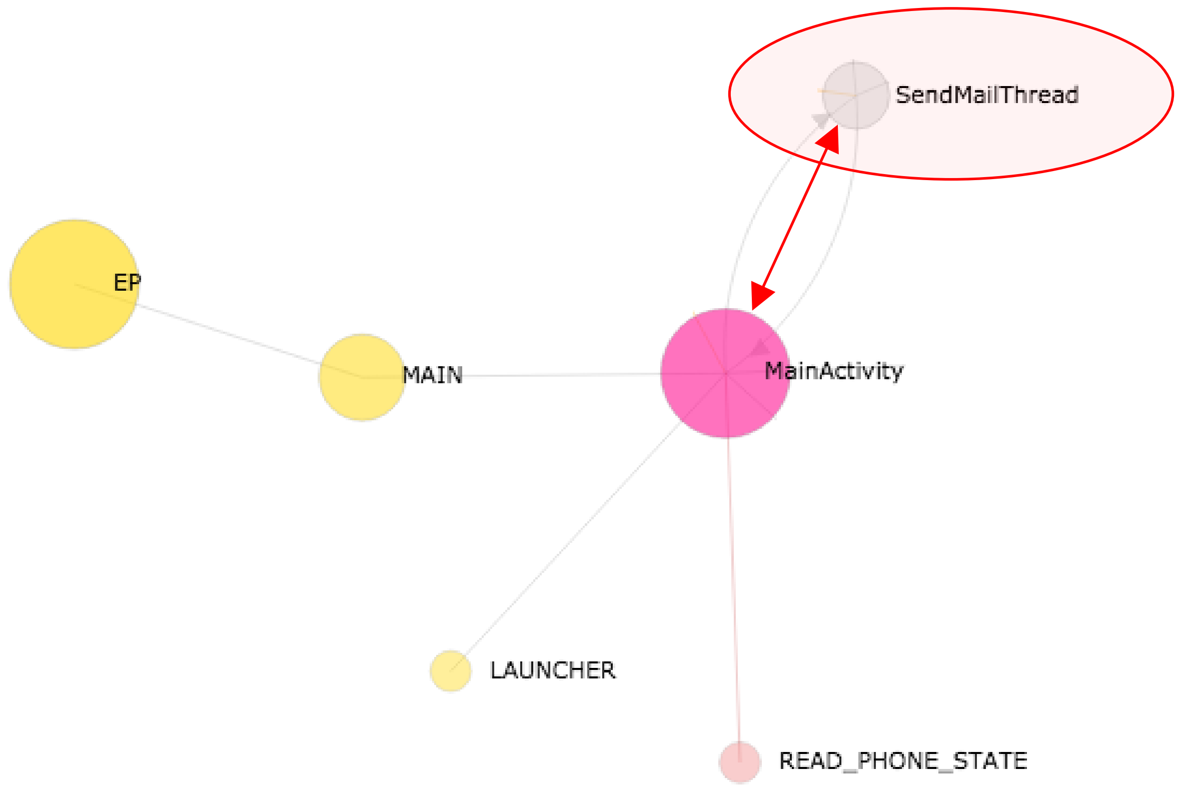
- ❖ reading AndroidManifest.xml
- ❖ classes inheriting Activity, Service, ContentProviders and BroadcastReceiver



Runnable Component (5/7)

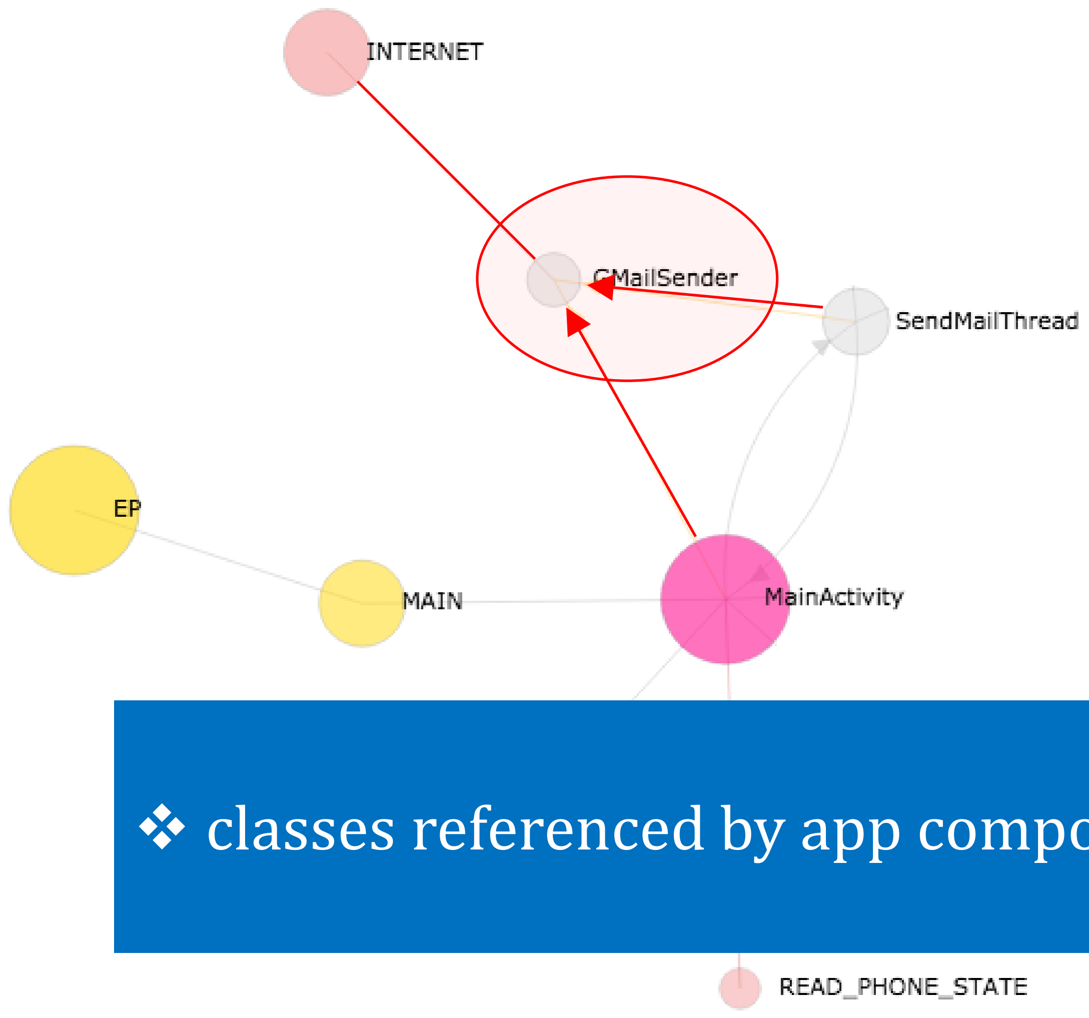
Android-Test/PNStealer

❖ classes inheriting Thread, Runnable and AsycTask



Import (6/7)

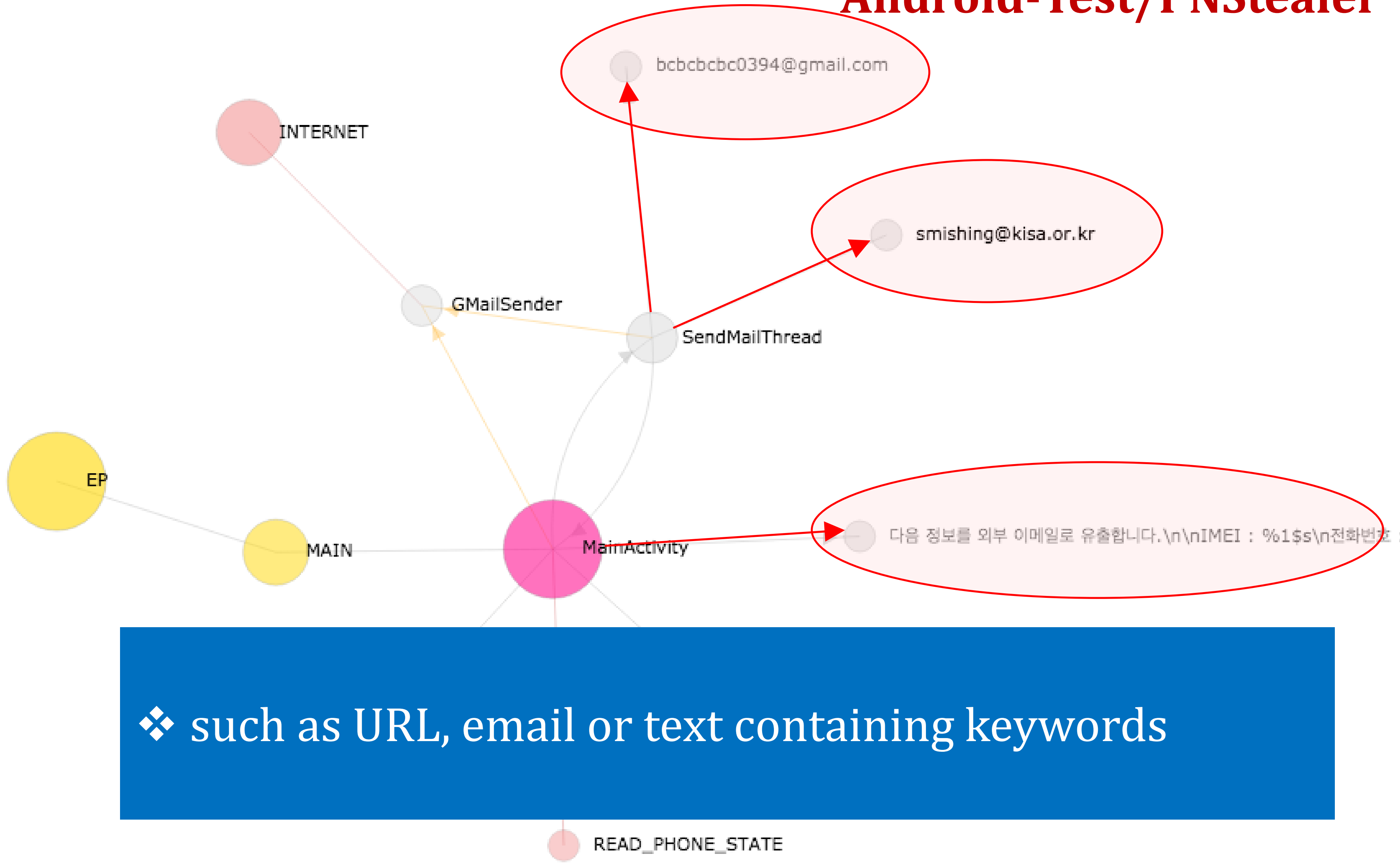
Android-Test/PNStealer



❖ classes referenced by app components or imports

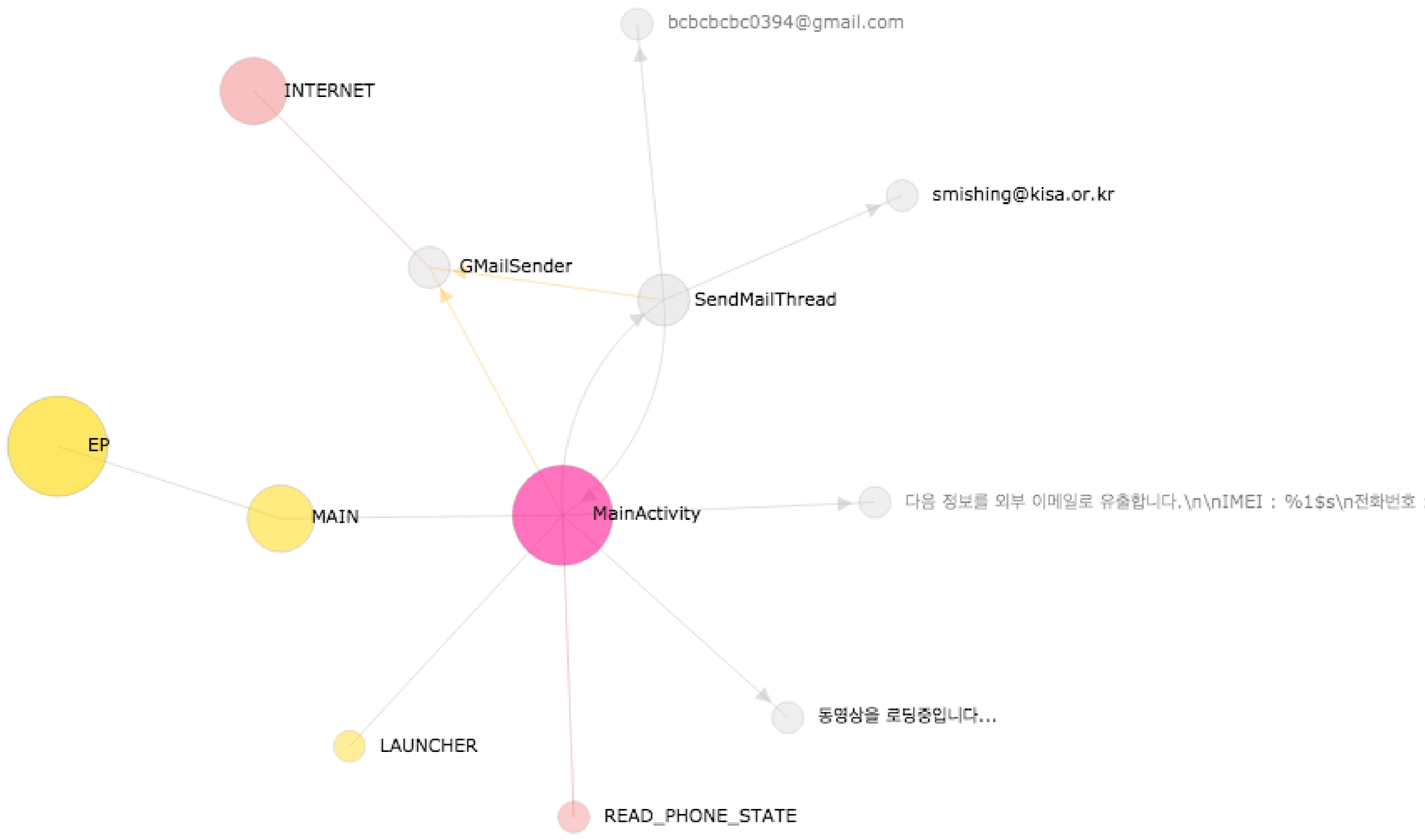
String (7/7)

Android-Test/PNStealer



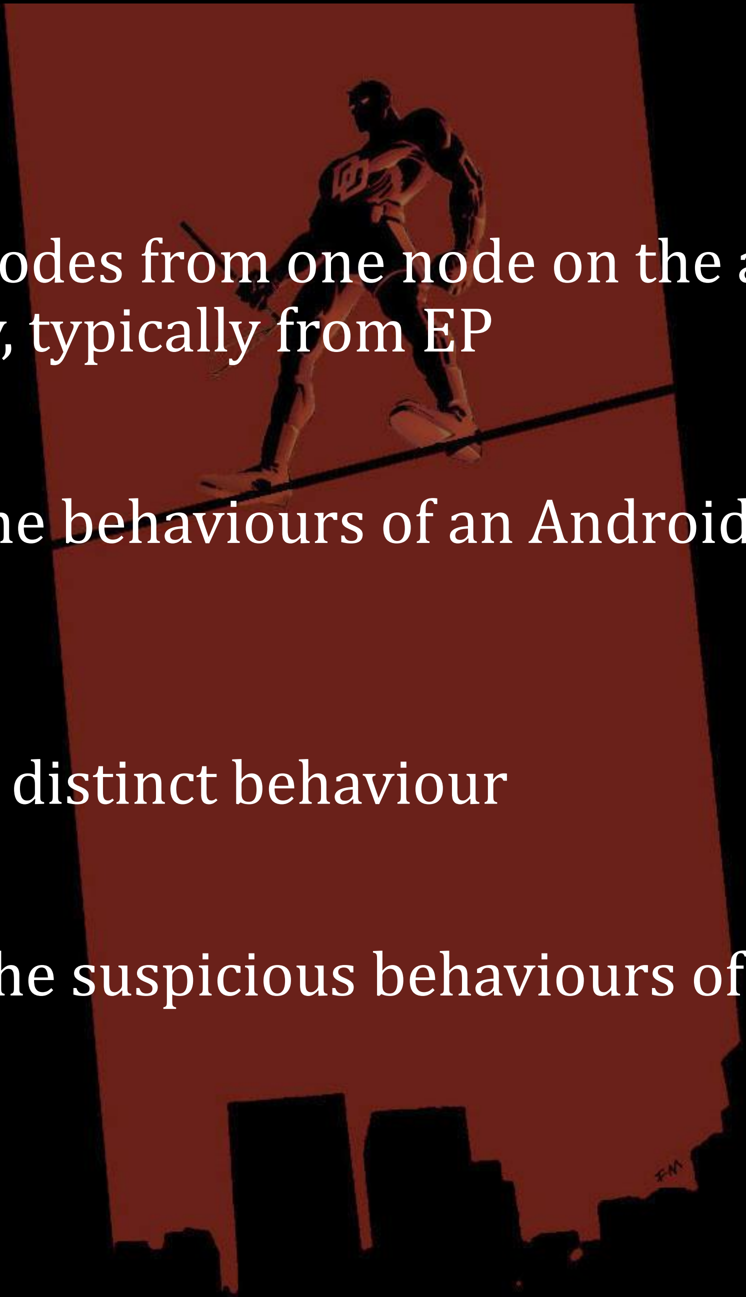
❖ such as URL, email or text containing keywords

A Complete Graph

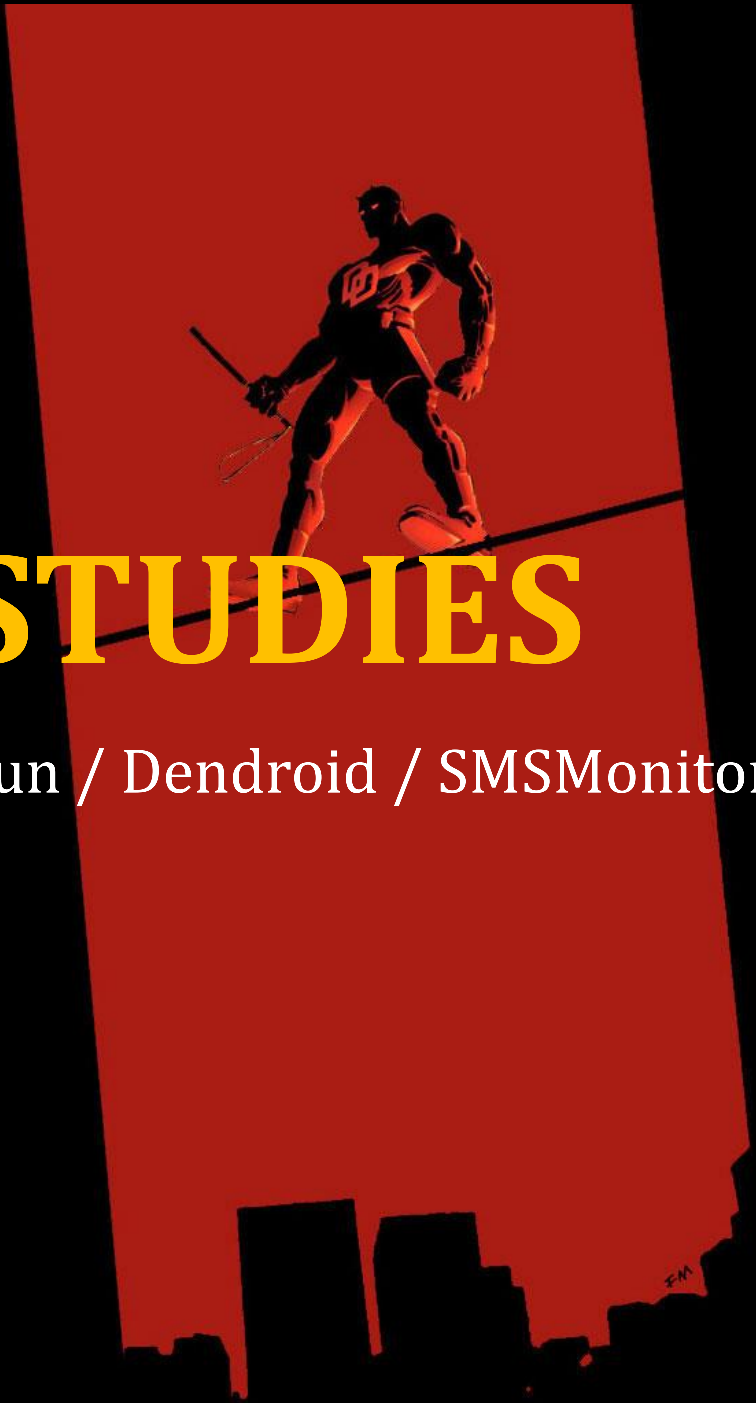


APP LIFECYCLING

App Lifecycling



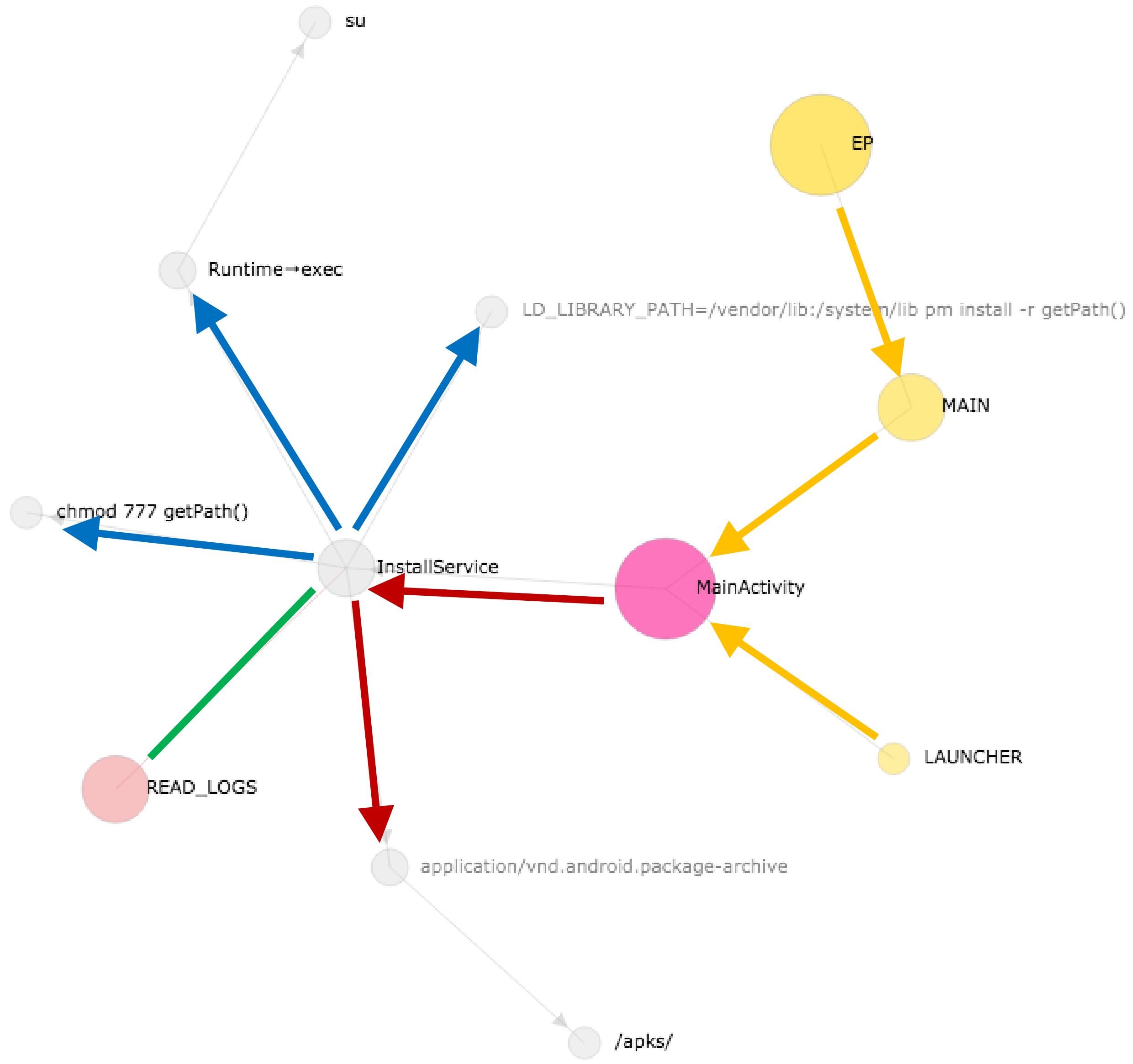
- ❖ Traversing all outgoing nodes from one node on the app lifecycle graph recursively, typically from EP
- ❖ useful for investigating the behaviours of an Android app
- ❖ effective for identifying a distinct behaviour
- ❖ well suited to detecting the suspicious behaviours of Android malware



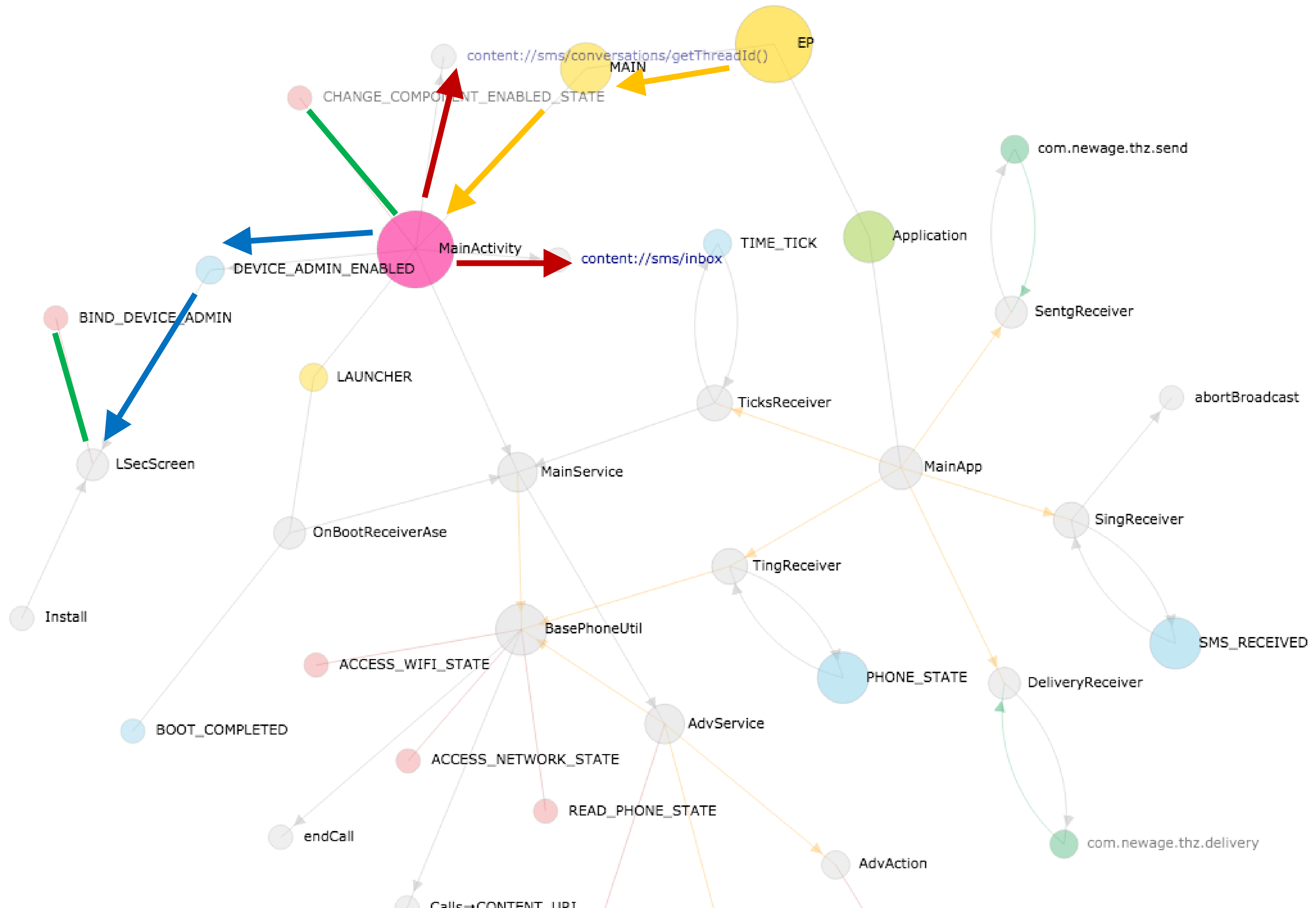
CASE STUDIES

Narut / KorTalk / Bankun / Dendroid / SMSMonitor

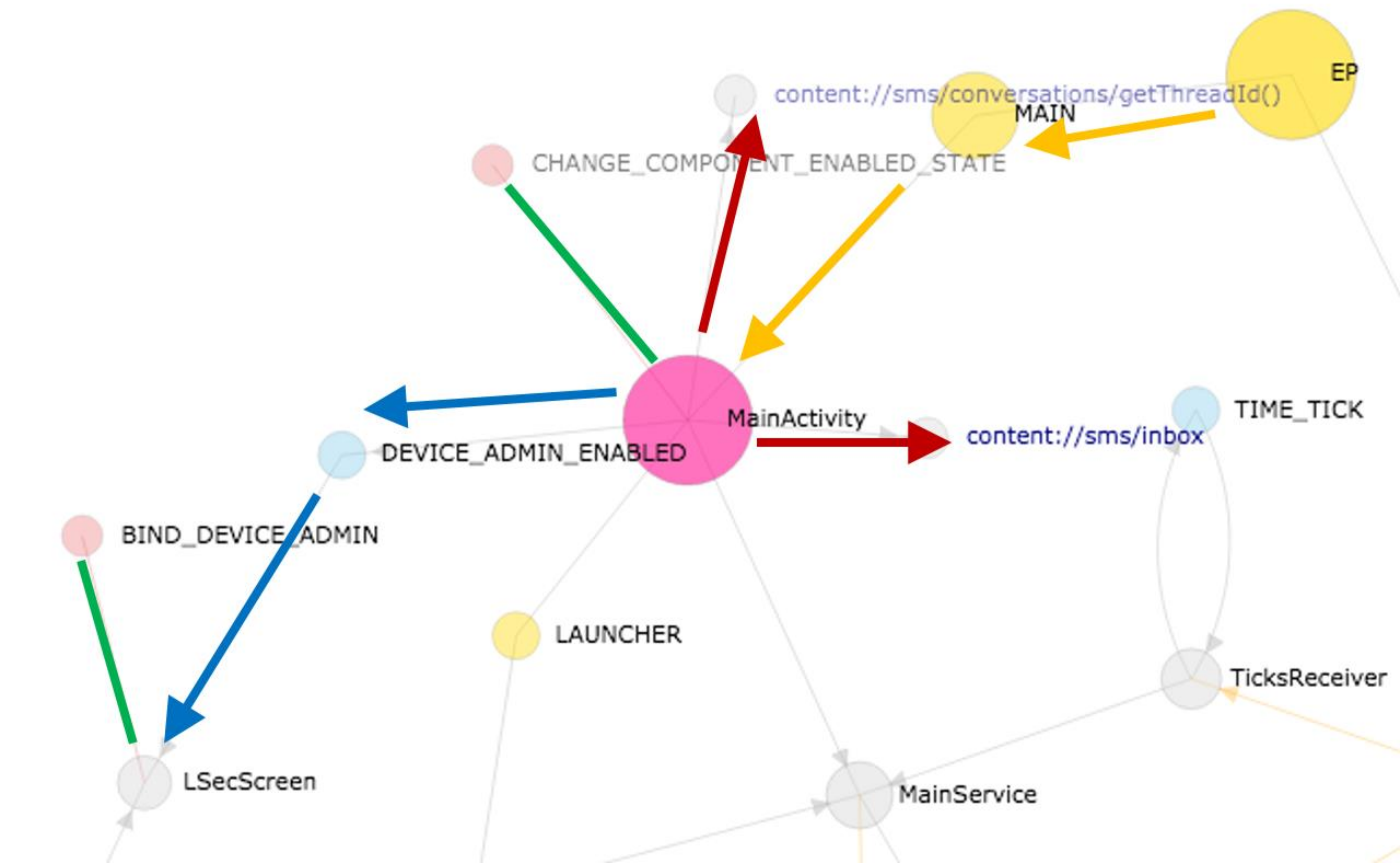
Trojan/Narut


su
EP
Runtime→exec
LD_LIBRARY_PATH=/vendor/lib:/system/lib pm install -r getPath()
MAIN
chmod 777 getPath()
InstallService
MainActivity
LAUNCHER
READ_LOGS
application/vnd.android.package-archive
/apks/

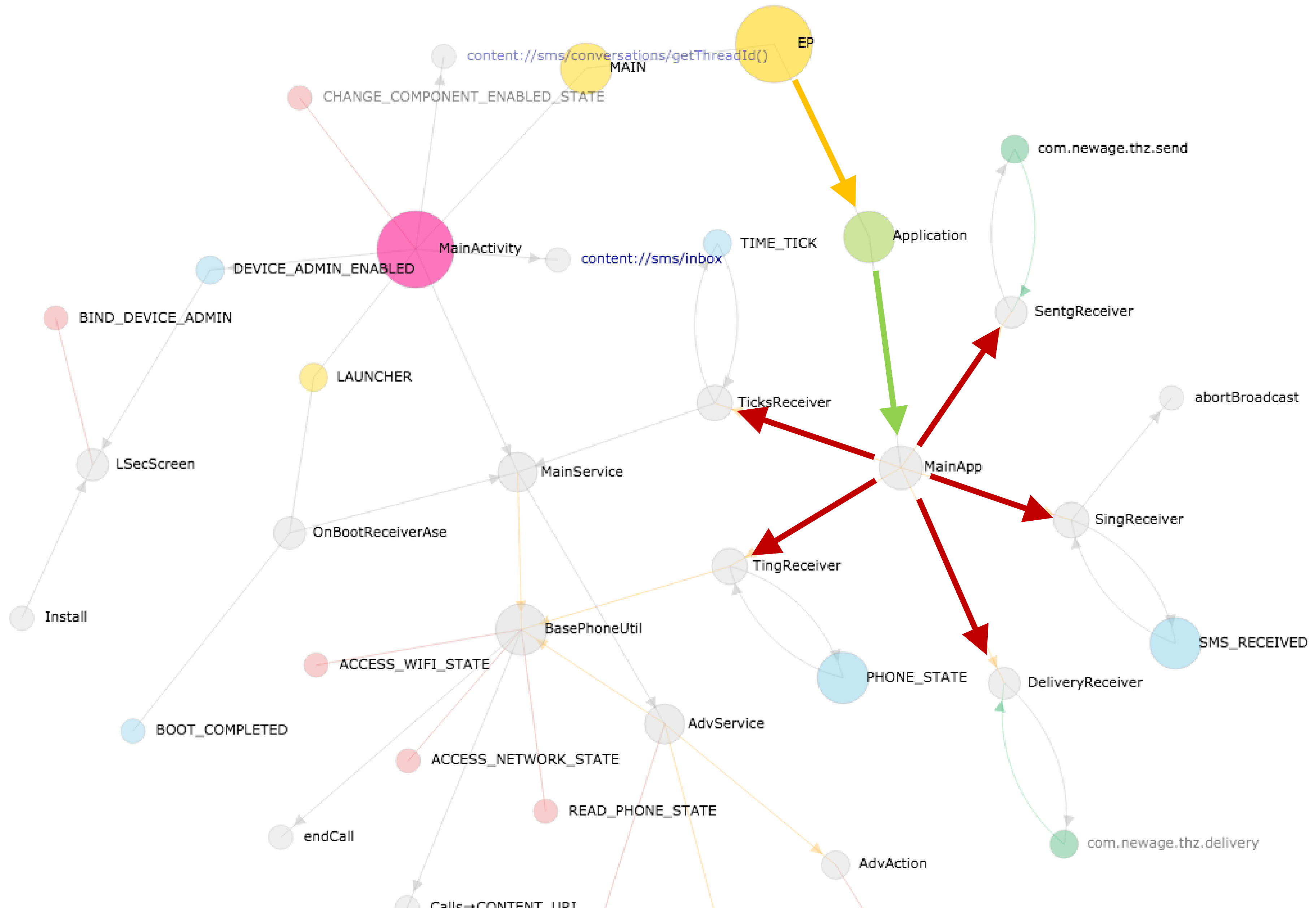
Trojan/KorTalk



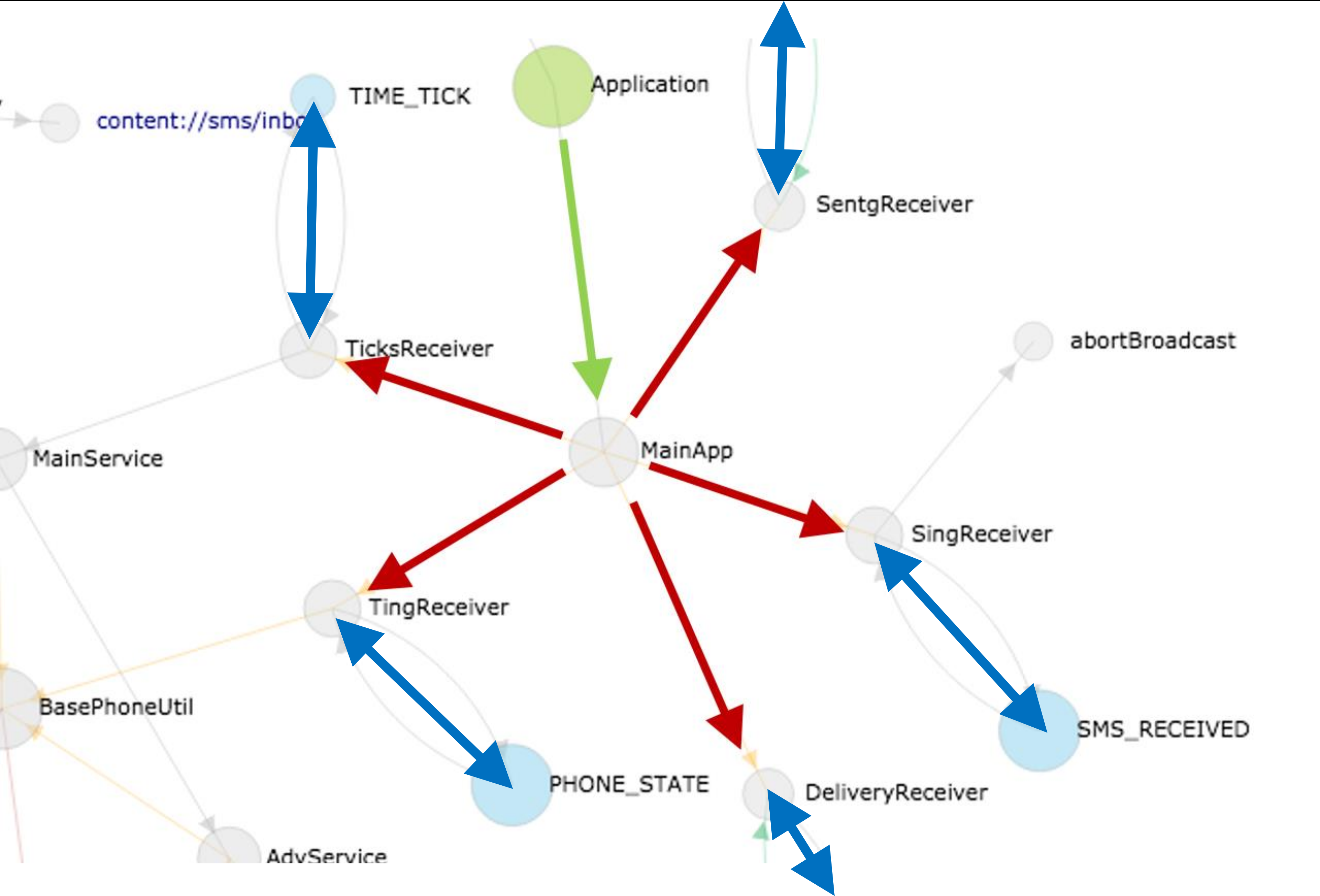
Trojan/KorTalk



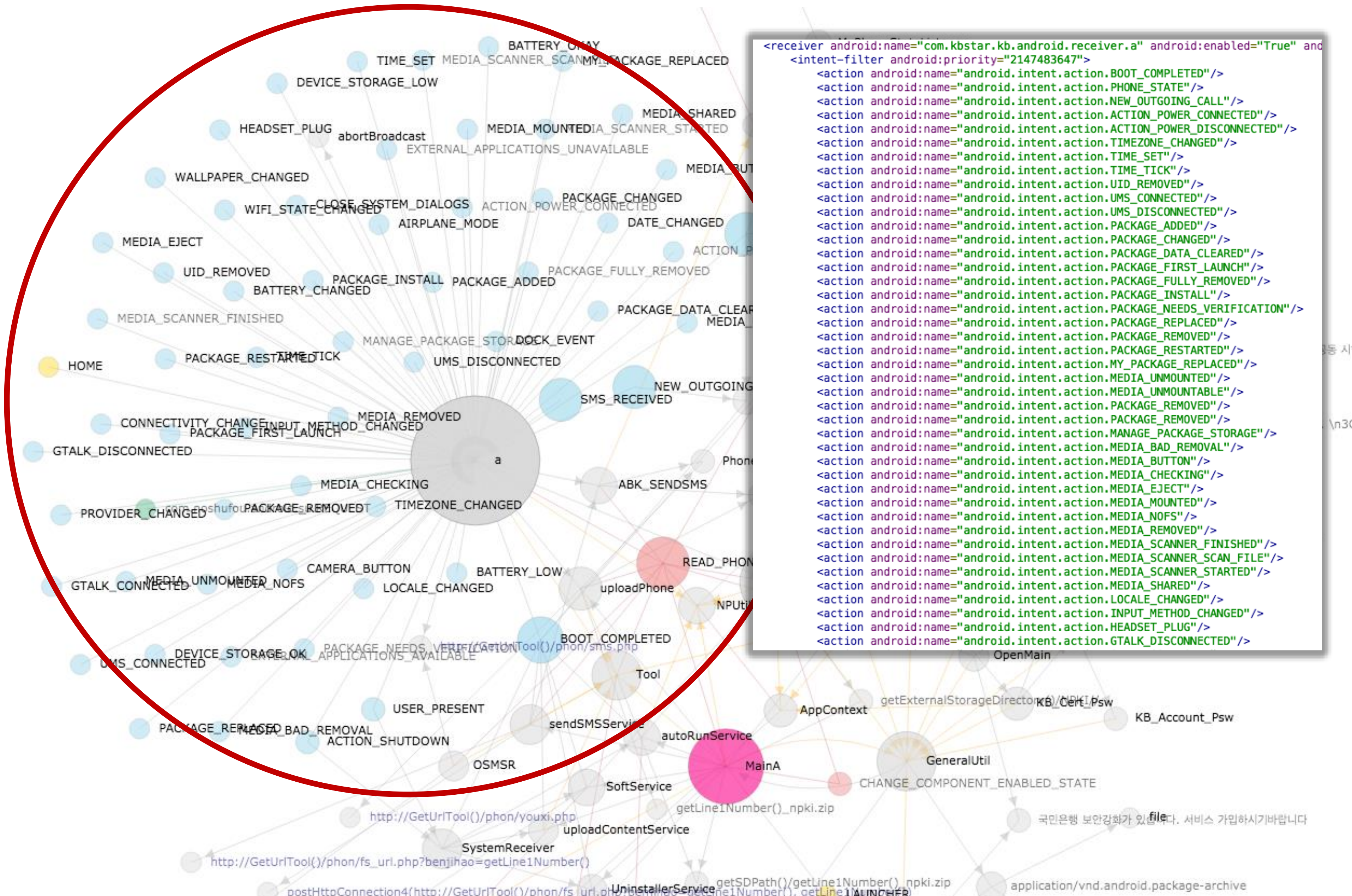
Trojan/KorTalk

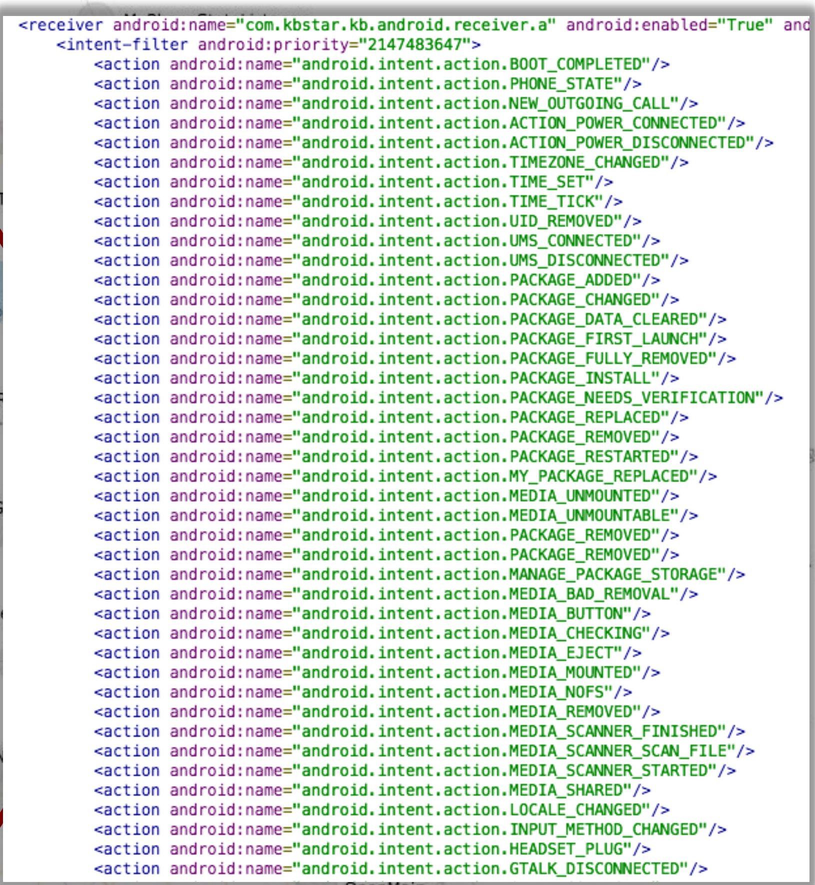


Trojan/KorTalk



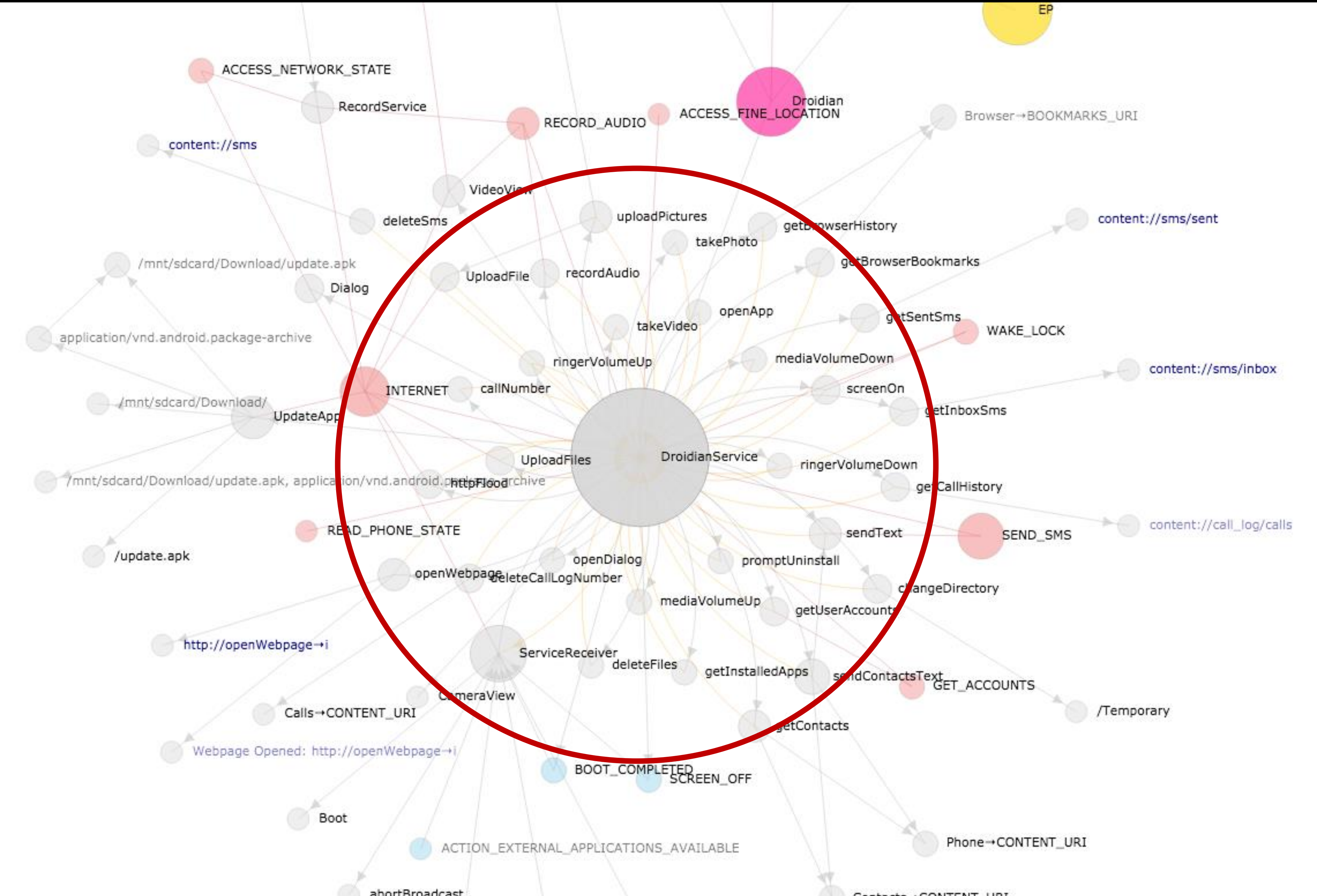
Trojan/Bankun



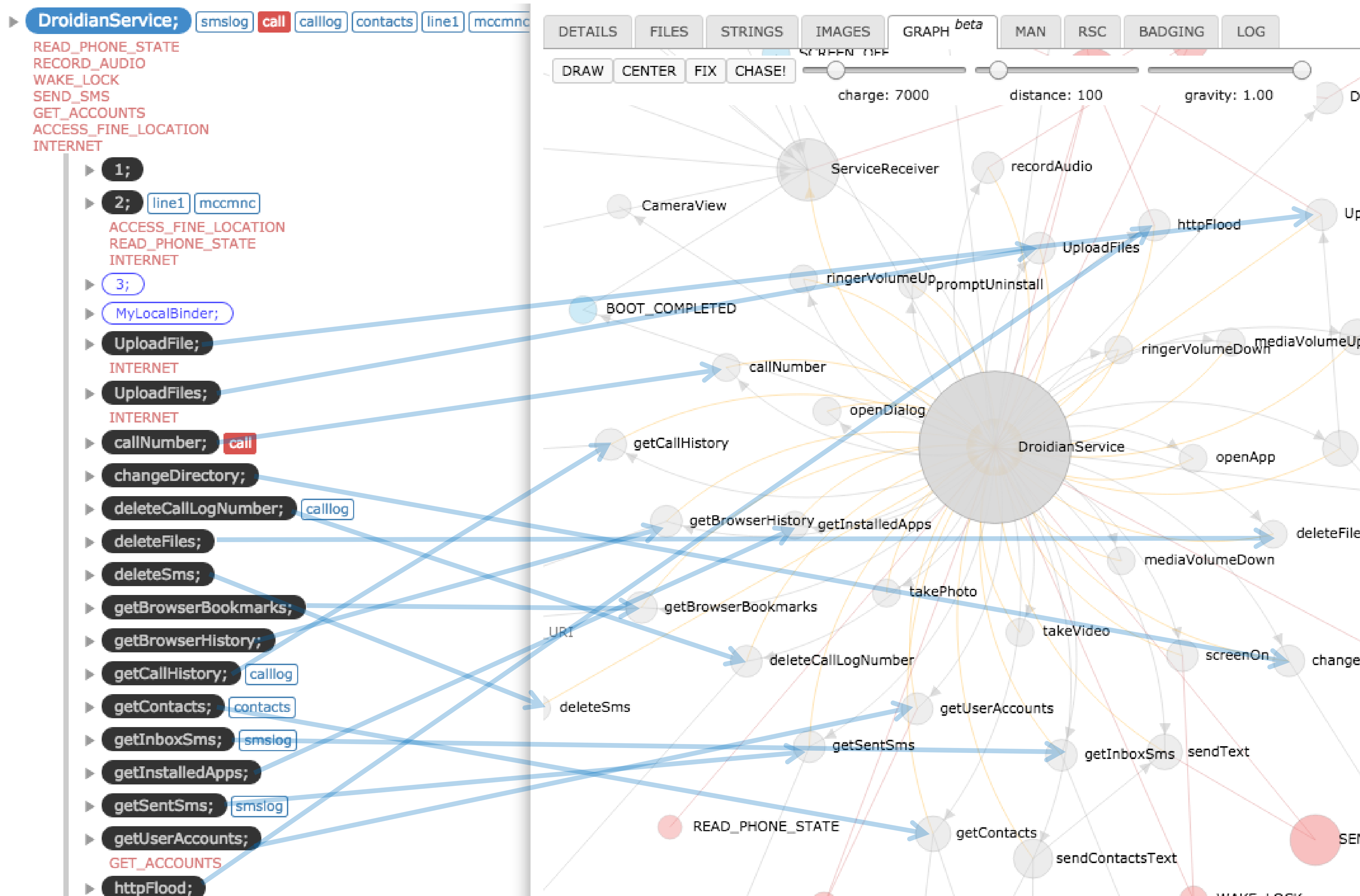


```
<receiver android:name="com.kbstar.kb.android.receiver.a" android:enabled="True" and
    <intent-filter android:priority="2147483647">
        <action android:name="android.intent.action.BOOT_COMPLETED"/>
        <action android:name="android.intent.action.PHONE_STATE"/>
        <action android:name="android.intent.action.NEW_OUTGOING_CALL"/>
        <action android:name="android.intent.action.ACTION_POWER_CONNECTED"/>
        <action android:name="android.intent.action.ACTION_POWER_DISCONNECTED"/>
        <action android:name="android.intent.action.TIMEZONE_CHANGED"/>
        <action android:name="android.intent.action.TIME_SET"/>
        <action android:name="android.intent.action.TIME_TICK"/>
        <action android:name="android.intent.action.UID_REMOVED"/>
        <action android:name="android.intent.action.UMS_CONNECTED"/>
        <action android:name="android.intent.action.UMS_DISCONNECTED"/>
        <action android:name="android.intent.action.PACKAGE_ADDED"/>
        <action android:name="android.intent.action.PACKAGE_CHANGED"/>
        <action android:name="android.intent.action.PACKAGE_DATA_CLEARED"/>
        <action android:name="android.intent.action.PACKAGE_FIRST_LAUNCH"/>
        <action android:name="android.intent.action.PACKAGE_FULLY_REMOVED"/>
        <action android:name="android.intent.action.PACKAGE_INSTALL"/>
        <action android:name="android.intent.action.PACKAGE_NEEDS_VERIFICATION"/>
        <action android:name="android.intent.action.PACKAGE_REPLACED"/>
        <action android:name="android.intent.action.PACKAGE_REMOVED"/>
        <action android:name="android.intent.action.PACKAGE_RESTARTED"/>
        <action android:name="android.intent.action.MY_PACKAGE_REPLACED"/>
        <action android:name="android.intent.action.MEDIA_UNMOUNTED"/>
        <action android:name="android.intent.action.MEDIA_UNMOUNTABLE"/>
        <action android:name="android.intent.action.PACKAGE_REMOVED"/>
        <action android:name="android.intent.action.PACKAGE_REMOVED"/>
        <action android:name="android.intent.action.MANAGE_PACKAGE_STORAGE"/>
        <action android:name="android.intent.action.MEDIA_BAD_REMOVAL"/>
        <action android:name="android.intent.action.MEDIA_BUTTON"/>
        <action android:name="android.intent.action.MEDIA_CHECKING"/>
        <action android:name="android.intent.action.MEDIA_EJECT"/>
        <action android:name="android.intent.action.MEDIA_MOUNTED"/>
        <action android:name="android.intent.action.MEDIA_NOFS"/>
        <action android:name="android.intent.action.MEDIA_REMOVED"/>
        <action android:name="android.intent.action.MEDIA_SCANNER_FINISHED"/>
        <action android:name="android.intent.action.MEDIA_SCANNER_SCAN_FILE"/>
        <action android:name="android.intent.action.MEDIA_SCANNER_STARTED"/>
        <action android:name="android.intent.action.MEDIA_SHARED"/>
        <action android:name="android.intent.action.LOCALE_CHANGED"/>
        <action android:name="android.intent.action.INPUT_METHOD_CHANGED"/>
        <action android:name="android.intent.action.HEADSET_PLUG"/>
        <action android:name="android.intent.action.GTALK_DISCONNECTED"/>
```

Trojan/Dendroid



Inner Class

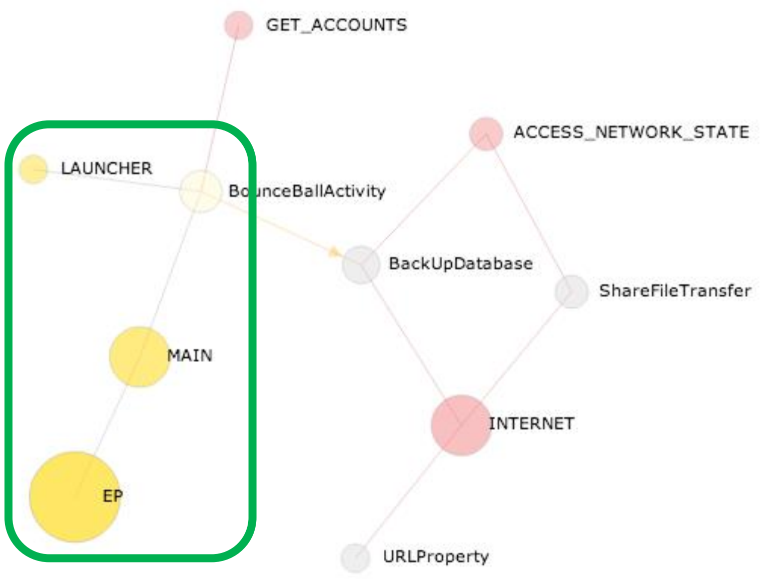

DroidianService; smslog call calllog contacts line1 mccmnc
READ_PHONE_STATE
RECORD_AUDIO
WAKE_LOCK
SEND_SMS
GET_ACCOUNTS
ACCESS_FINE_LOCATION
INTERNET

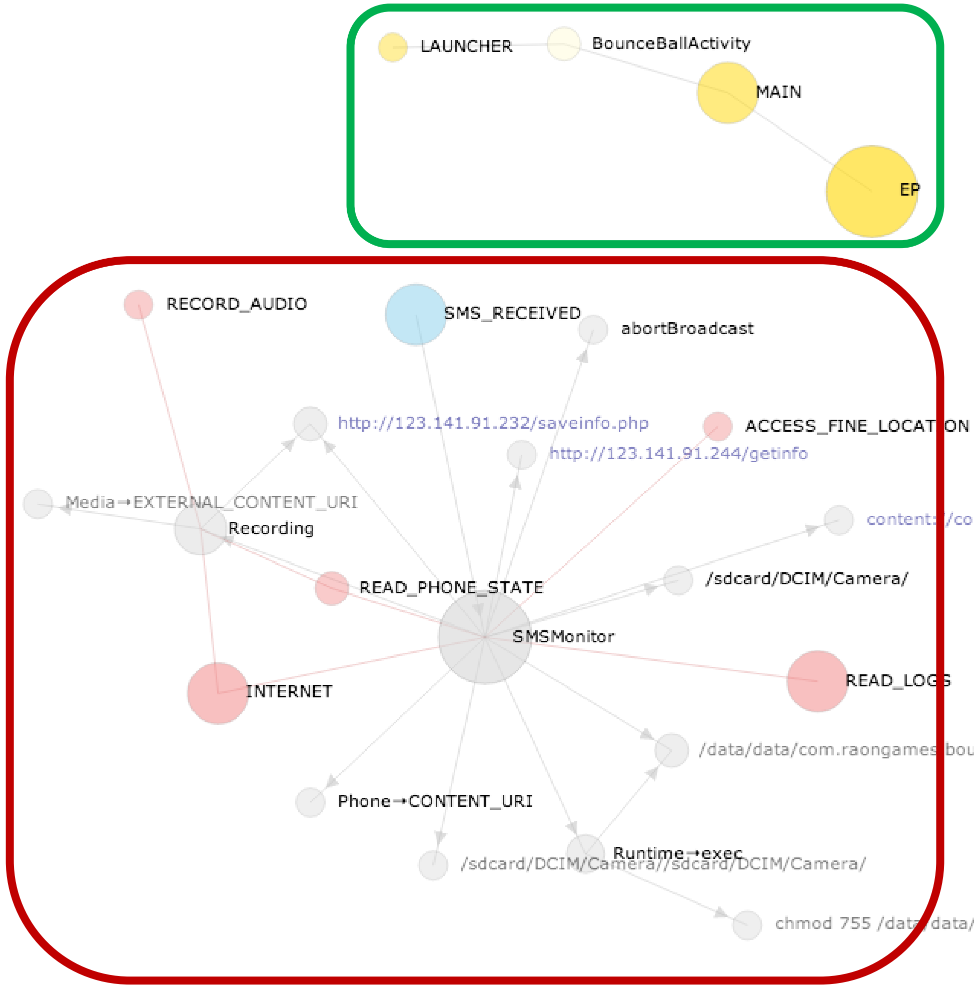
- 1;
- 2; line1 mccmnc
  ACCESS_FINE_LOCATION
  READ_PHONE_STATE
  INTERNET
- 3;
- MyLocalBinder;
- UploadFile;
  INTERNET
- UploadFiles;
  INTERNET
- callNumber; call
- changeDirectory;
- deleteCallLogNumber; calllog
- deleteFiles;
- deleteSms;
- getBrowserBookmarks;
- getBrowserHistory;
- getCallHistory; calllog
- getContacts; contacts
- getInboxSms; smslog
- getInstalledApps;
- getSentSms; smslog
- getUserAccounts;
  GET_ACCOUNTS
- httpFlood;

Repackaging

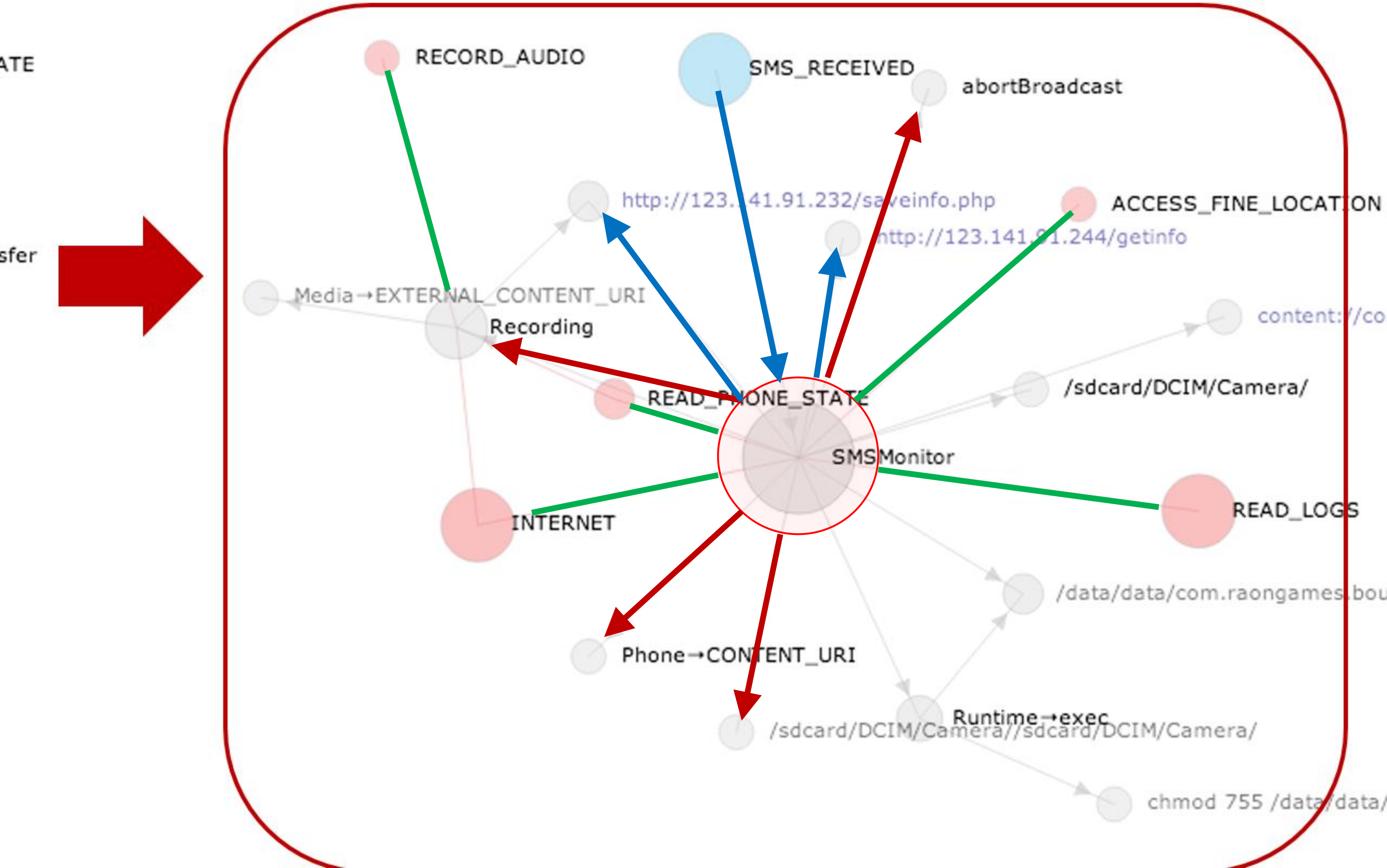
BounceBall

Android-Backdoor/SMSMonitor

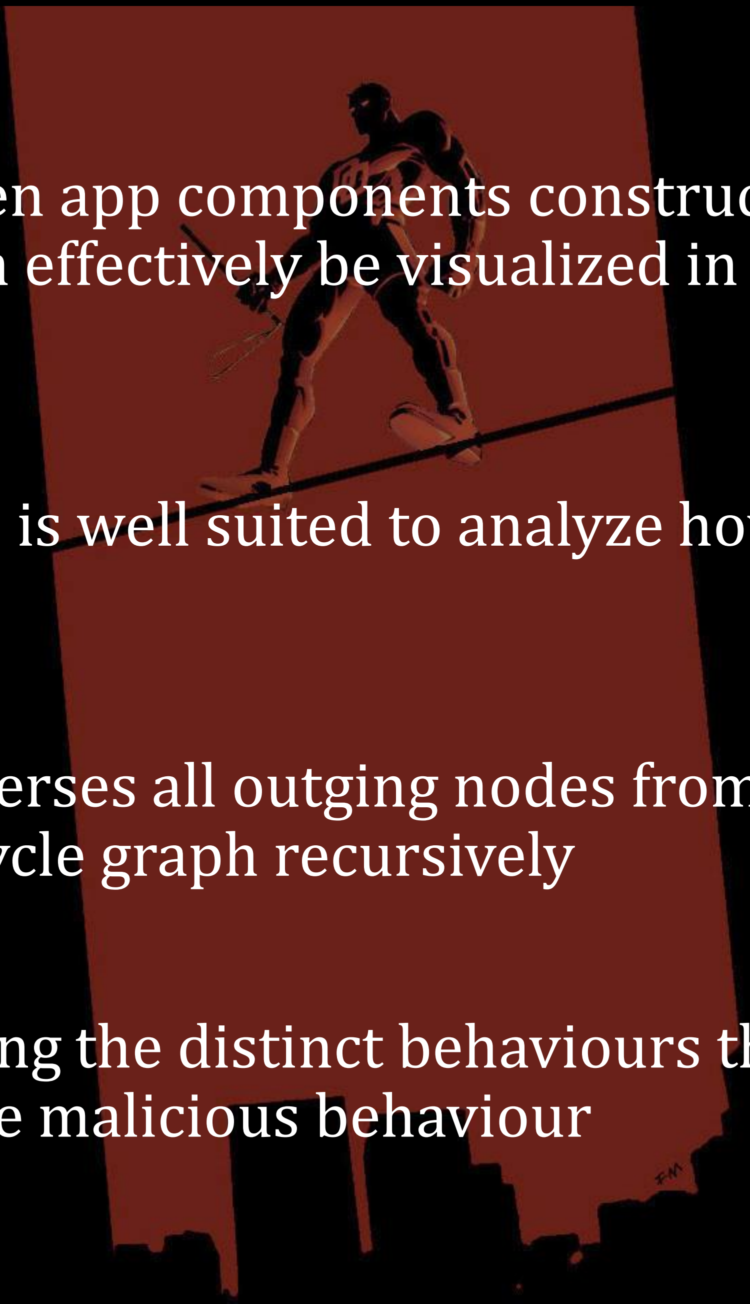




Repackaging



Conclusions



❖ The relationships between app components construct the ***App Lifecycle***, and can effectively be visualized in a graph

❖ The ***App Lifecycle Graph*** is well suited to analyze how an Android app operates

❖ The ***App Lifecycling*** traverses all outging nodes from one node on the app lifecycle graph recursively

❖ is so effective in identifying the distinct behaviours that it can be used to detect the malicious behaviour

Thank you

jypark@ahnlab.com