Detection Of Trojan In Android Devices

Manuel Antony
Guided By
Dr.Prabhaker Mateti

Agenda

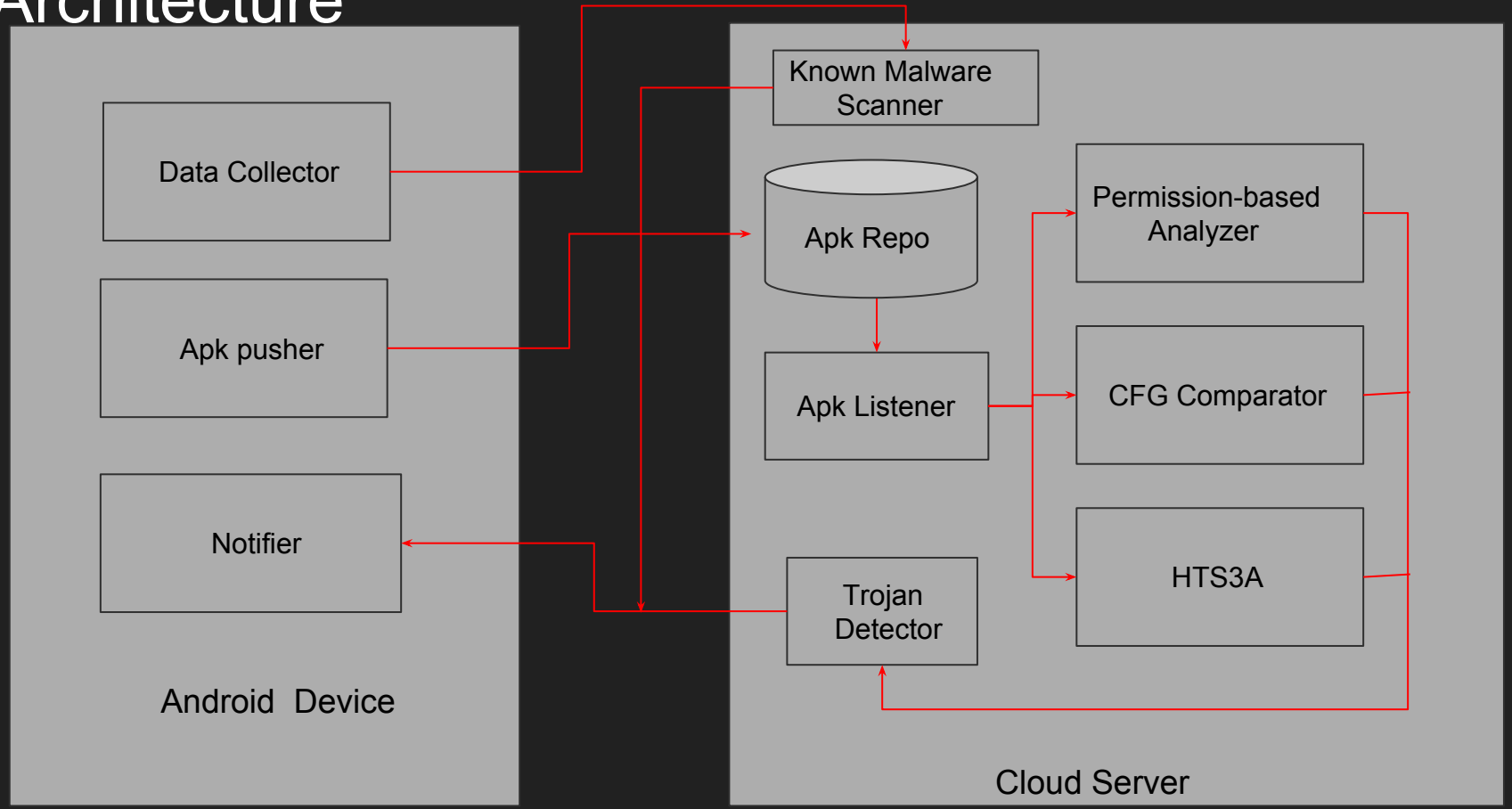
- Problem Statement
- Architecture
- Importance of our approach
- Restricted model
- Approaches in detecting Trojan:
    - Permission-based analyzer
    - CFG Comparator
    - HS3A
- Trojan detection
- Work done till now :
    - Implementation of method 1

Problem Statement

- To create an integrated framework that analyzes apps installed in user's Android devices and further notify users if the app has characteristics of a Trojan malware.

- Methodology:
    - Static and dynamic analysis

Architecture



Architecture.

- Data collector (in device):
    - Collects :
        - Fully qualified package name
        - MD5 hash
        - SHA 1 hash
- Known Malware Scanner (in cloud):
    - Hashes checked against a black-list of malicious APKs
    - Fully qualified package name and hashes

From minor Project

Architecture..

- APK Pusher: push app into the cloud server
- APK Repo: storage for APK in cloud
- APK Listener: read the apk name and path of the APK
- Trojan detector:
    - Aggregate the results from method 1, method 2 and method3
    - Analyze the results
    - Determine the APK is a Trojan or not.
    - Notify the Notifier in the users device
- Notifier: notify the user that a particular app is a Trojan malware

Contributions of our approach

- Framework that offers bility to analyze any application the users installs.
- Integrates several static and dynamic analysis techniques
- Trojan Detection As A Service offered to the user.
- Unlike current approaches :
    - Every analysis is done on cloud server
    - Less processor overhead for user’s device
    - Minimal use of user's memory resources

Manual analysis

- More efficient
- Possible to analyse :
    - Line by line code by reversing
    - Use tools like wireshark for understanding the network analysis
    - Humans can put more instinct towards detection
- Not scalable :
    - Difficult to analysis N number of application real time

Restricted model Behavioral analysis of Trojan

- What is restricted model?
- Why a restricted model?
- What are the behaviours which is analyzable?

What is a Restricted model?

- Model based on parameters which is fixed
- Parameters can be:
    - Any of the observable values
    - Results of a test

Why is a restricted model?

- Automated analysis is bound to certain limits
- This is due to :
    - Dynamic analysis is limited to tools
    - Static analysis is limited to data analysed before
    - Automated analysis always follows a restricted model

Analyzable behaviour and data

- Automated analysis should be bound to certain parameters
- Parameters are:
    - Permissions
    - API calls
    - HTTP communication
    - Hashes
    - Package names
    - Logs
    - Payloads

Prolog

- After doing the analysis of package name and hashes the apk will be pushed into the cloud server.
- A Java API will listen for this.

Steps:

1. Using Java API, register a folder for apk
2. If a apk is pushed to that folder, it will take that apk for analysis

Implementation Language, Tools

Tools : Java 8, apktool, intellij idea

Apktool : For decompiling .apk file

Language : Java 8

Java JDK version : openJDK 1.8

Cloud : Amazon Web Service

Prolog..

1. Using Java API, register a folder for apk :

    addListenrToADirectory(“Directory Name”){

        return true if a new package is pushed to this directory }

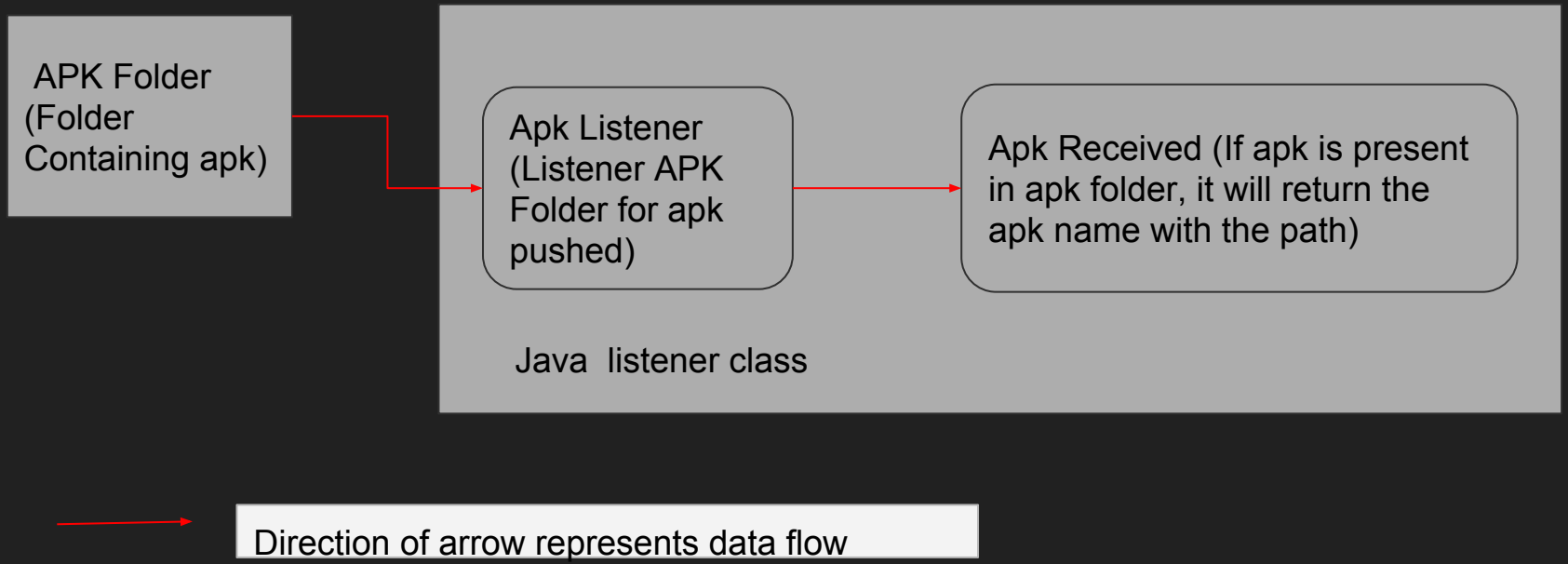
2. If a apk is pushed to that folder it will take that apk for analysis

    boolean apkRecived = addListenrToADirectory(“Directory Name”)

    if (apkRecived){

        return the apk name with path }

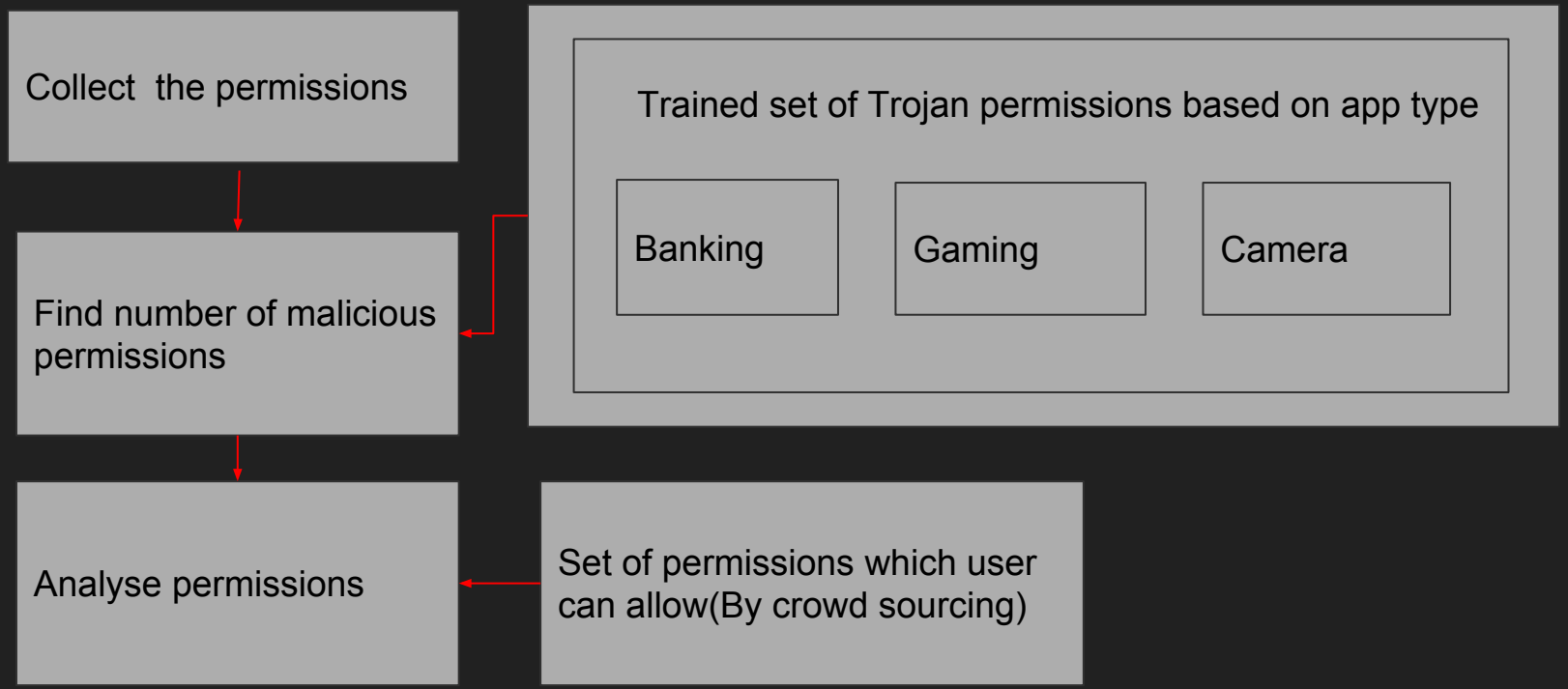
Prolog...


APK Folder (Folder Containing apk)
Apk Listener (Listener APK Folder for apk pushed)
Apk Received (If apk is present in apk folder, it will return the apk name with the path)
Java listener class
Direction of arrow represents data flow

Approaches

- Here we are approaching three methodologies, they are :
    - Permission-based Analyzer :
        - Permission based analysis
    - CFG (Control Flow Graph):
        - API calls
    - HS3A (Http/s Subtree Similarity Search Algorithm):
        - Http/s communications

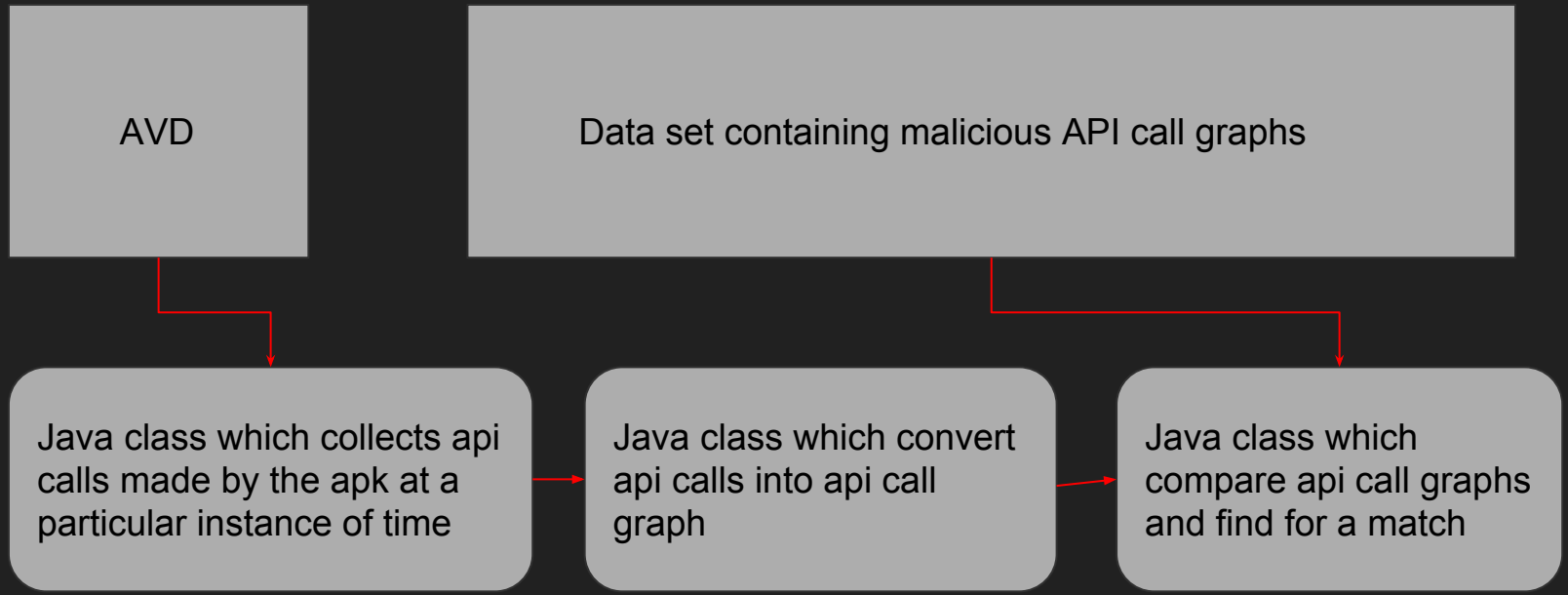
Permission based analyzer



Permission based analyzer ..

- Algorithm:
    - 1. If apk received
        - 1.1. Collect the permissions
        - 1.2. Compare collected permissions based trained category of malicious permissions
        - 1.3. Eliminate the permissions, which user allowed, from the result of compared permissions
        - 1.4. Divide the number of permissions obtained in 1.3 with malicious permissions collected in 1.2 based on category; results in value <= 1
- Step 1.4 will give a threshold of Trojan behaviour :
    - If value < 0.7 : strictly Trojan
    - Else if value > 0.3 && <= 0.7 : Intermediate Trojan
    - Else if value <= 0.3 : Not a Trojan

CFG


AVD
Data set containing malicious API call graphs
Java class which collects api calls made by the apk at a particular instance of time
Java class which convert api calls into api call graph
Java class which compare api call graphs and find for a match

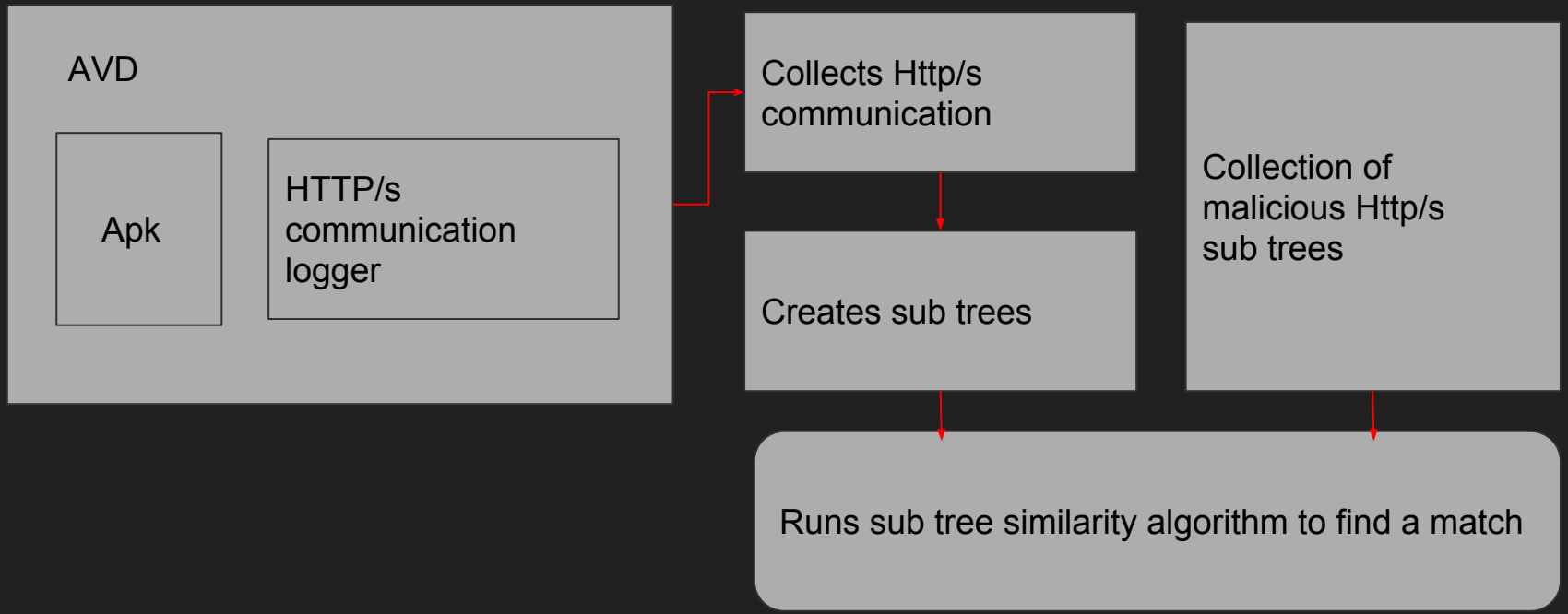
CFG : Collecting API calls

- Following are the steps for collecting api calls:
    - Install apk in an AVD
    - Invoke adb shell
    - Start the app using command : am start
    - List the running process using command : ps
    - Find the process ID related to the app
    - Start profiling by using command : START /b adb shell am profile pid start
- Profiling process execution will allow us to:
    - monitor and record the details on its execution
    - including what methods are invoked and what resources are being utilized at which times

CFG : cont..

- Algorithm:
    - 1. Collects the api calls
    - 2. Create api call graph
    - 3. Compare api call graph with trained api call graph
    - 4. Based on comparison categories the apk as :
        - If complete match : Strictly Trojan
        - Else If match is intermediate : Intermediate Trojan
        - Else If no match : Not a Trojan

HS3A


AVD
Apk
HTTP/s communication logger
Collects Http/s communication
Creates sub trees
Collection of malicious Http/s sub trees
Runs sub tree similarity algorithm to find a match

HS3A : cont..

- Algorithm :
    - 1. Runs the apk
    - 2. Collects http/s communication within a time period
    - 3. Create trees based on http/s communication
    - 4. Runs subtree similarity search algorithm against trained set of sub trees
    - Look a match found :
        - If it is complete match : Strictly Trojan
        - Else if match is intermediate : Intermediate Trojan
        - Else : Not a Trojan

Strictly a Trojan

- Based on explained approached we can say a apk is strictly Trojan if any of the following results appear :

Permission : Strictly Trojan
Api call graph : Strictly Trojan
Http/s : Strictly Trojan

Permission : Strictly Trojan
Api call graph : --
Http/s : --

Permission : --
Api call graph : Strictly Trojan
Http/s : --

Permission : --
Api call graph : --
Http/s : Strictly Trojan

Syntax :
<Type of Method> : <Result Obtained>

-- can be any result

It can be a Trojan

- Based on explained approached we can say a apk can be Trojan if any of the following results appear :

Permission : Intermediate Trojan
Api call graph : Intermediate Trojan
Http/s : Intermediate Trojan

Permission : Intermediate Trojan
Api call graph : --
Http/s : --

Permission : --
Api call graph : Intermediate Trojan
Http/s : --

Permission : --
Api call graph : --
Http/s : Intermediate Trojan

-- can be any result

Not a Trojan

- Only one result concludes the apk is not a Trojan :

Permission : Not a Trojan
Api call graph : Not a Trojan
Http/s : Not a Trojan

Work done till now

- Method 1
- Trained data sets by manual analysis

Implementation of Method 1

- Technologies used:
  - Language : Java 8
  - IDE : intellij Idea
  - Platform : Linux Ubuntu 16.04 LTS
- Java classes :
  - ListnerInterface.class -- Interface for listener class, which will listen for package added in the registered folder
  - ApkToolDecom.class -- decompile the apk using apktool
  - PermissionFetch.class -- fetch every permission of an apk
  - RestrictedModelAnalysisPermission.class :
    - Do the comparison as suggested in method 1
    - Categorises app as :
      - Strictly a Trojan
      - Intermediate Trojan
      - Not a Trojan

THANK YOU