Detection Of Trojan In Android Devices

Manuel Antony
Guided By
Dr.Mateti

Preface

- What is Trojan malware?
- What a Trojan malware can do?
- How It is spread?
- Types of Trojan malware
- Significance of Trojan malware in Android
- Work done for minor project.

What we are trying to do?

- Real time analysis of Trojan malware in Android devices.

    - Analysis of APKs installed in user devices.

    - Analysis Using :

        - Static and dynamic analysis

Manual analysis

- Manual analysis is more efficient
- In manual analysis it is possible to analyse :
    - Line by line code by reversing
    - Use tools like wireshark for understanding the network analysis
    - Humans can put more instinct towards manual analysis

- But manual analysis is not scalable :
    - Difficult to analysis N number of application real time

Importance of our approach

- Analysis should scalable for N number of application at a particular time
- It is important that every analysis should be automated

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?

- 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

- Why a restricted model ?
- Due to :
    - Automated analysis is dependent on static and dynamic analysis
    - Automated analysis always follows a restricted model

Analysable behaviours and datas

- Unlike manual analysis it is impossible to analyse every details of an application by automated analysis.
- So the analysis should bound to certain parameters
- Parameters are:
    - Permissions
    - API calls
    - HTTP communication
    - Hashes
    - Package names
    - Logs
    - Payloads

Approaches

- Here we are approaching three methodologies, they are :
    - Approach 1 :
        - Permission based analysis
    - Approach 2 :
        - API calls
    - Approach 3 :
        - Http/s communications

Initial State

- 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

Initial State..

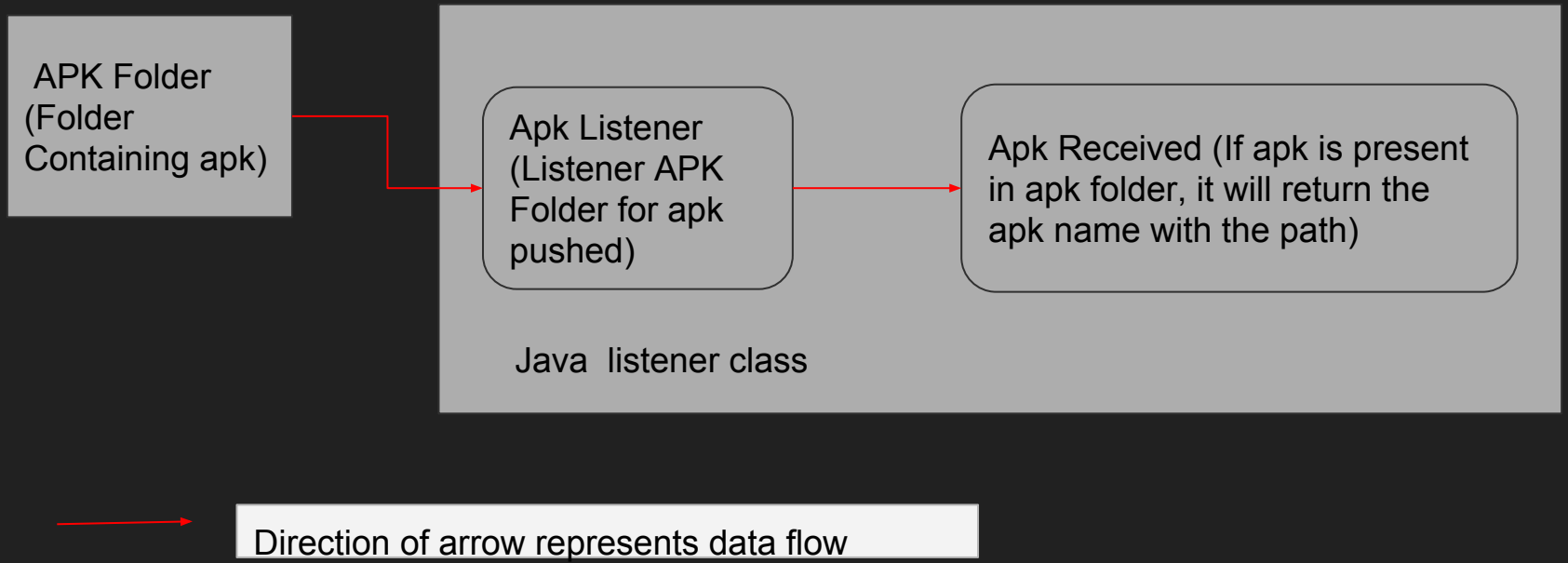
1. Using Java API, register a folder for apk :

```
addListenrToADirectory(“Directory Name”){
    return  true if a new package is pushed to this directory }
```

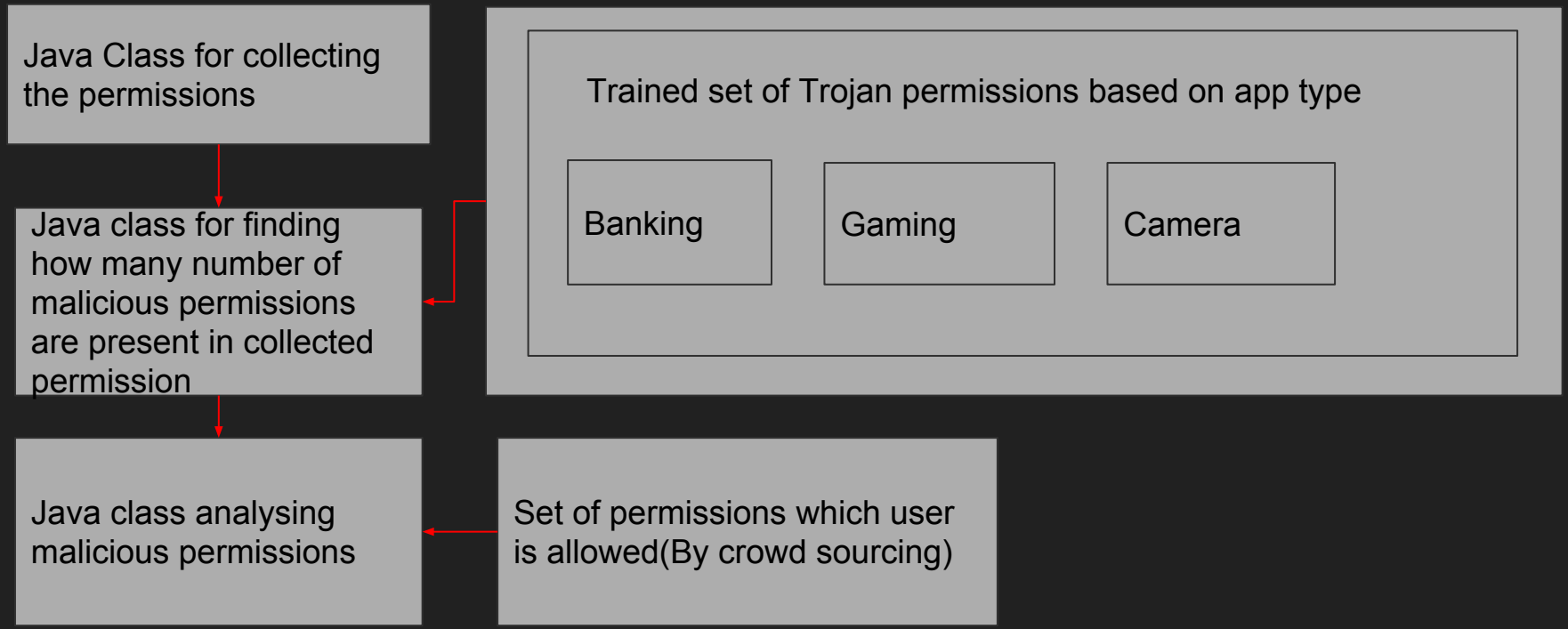
2. If a apk is pushed to that folder it will take that apk for analysis

```
boolean apkRecived = addListenrToADirectory(“Directory Name”)
if (apkRecived){
    return the apk name with path }
```

Initial state:



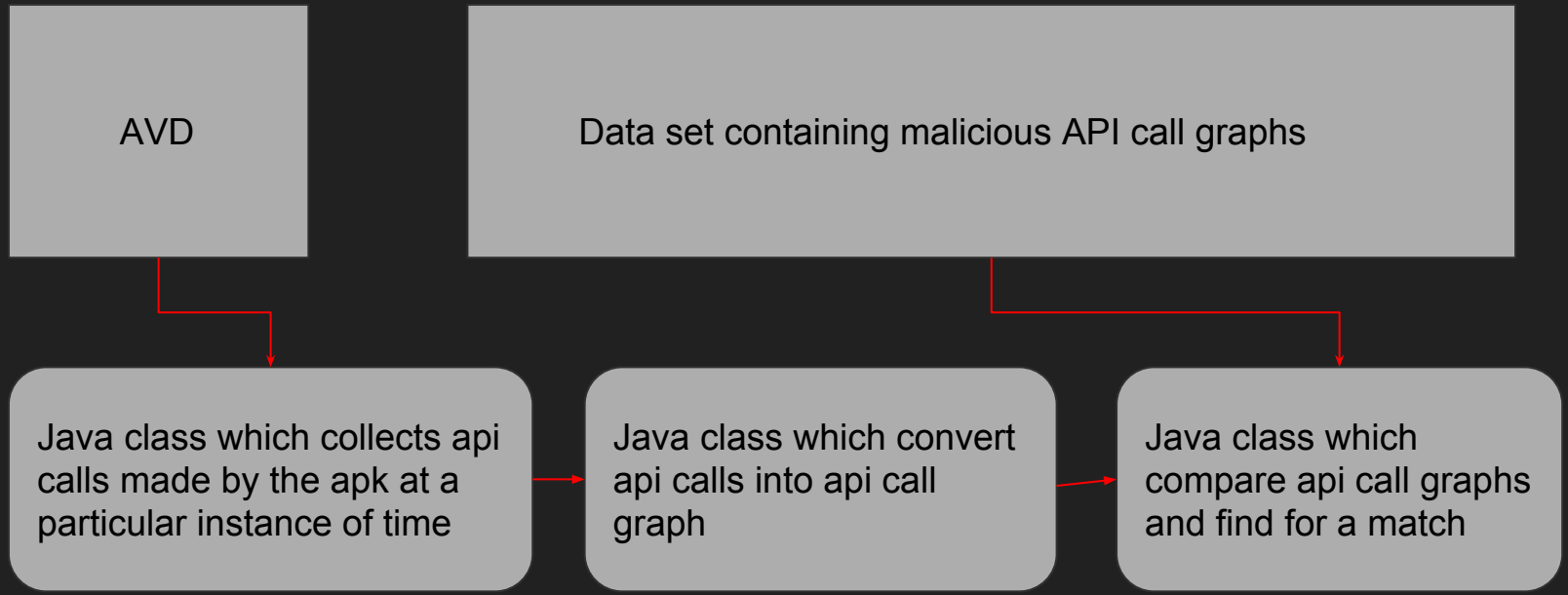
Approach 1 : Permission based analysis



Approach 1 : Permission based analysis...

- 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

Approach 2 : API Calls



Approach 2 : Collecting API calls

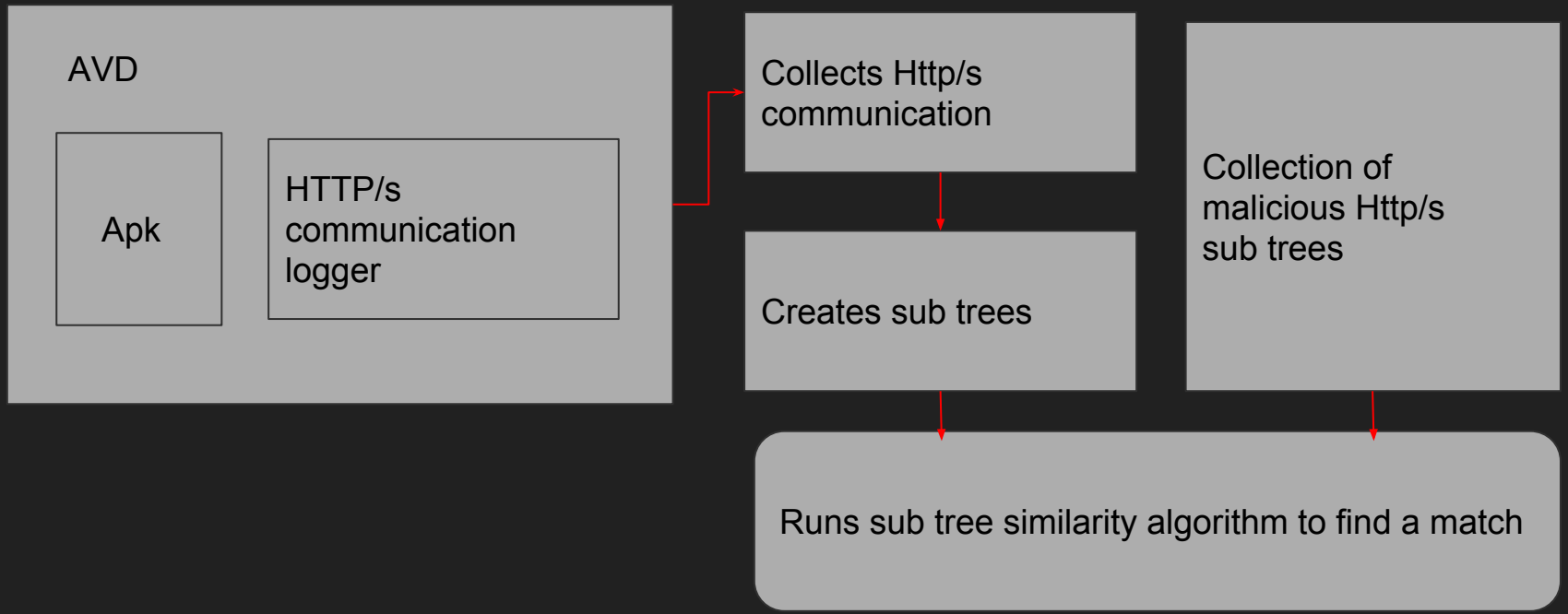
- Following are the steps for collecting api calls:
    - Install apk in an AVD
    - Invoke adb shell
    - Start the app using command : am start
    - List the running process using command : ps
    - Find the process id related to the app
    - Start profiling by using command : START /b adb shell 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

Approach 2 : 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

Approach 3 : Http/s communication



Approach 3 : 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
    - 5. 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 Approach> : <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

- Implemented method 1
- Trained data sets

THANK YOU