STDNeut: Neutralizing Sensor, Telephony System and Device State Information on Emulated Android Environments

Saurabh Kumar, Debadatta Mishra, Biswabandan Panda, and Sandeep K. Shukla
Indian Institute of Technology, Kanpur

Android Emulator

- Used for prototype develop and test an application
- Dynamic Analysis of malware
  - Run applications on an Emulator
  - Detect malicious behavior
- Problem:
  - Malware writers inserts emulation-detection code to evade dynamic analysis

Our Goal

Objective 1: Designing a emulation-detection library to study the efficacy of dynamic analysis framework

Objective 2: Developing an anti-emulation-detection platform to Neutralize sensors, telephony system and device state information

EmuDetLib: Emulation-Detection Library

- Detection methods are classified in 5 category
  - Unique Device Information (basic and **smart**)
  - Sensors Reading
  - GPS Information
  - Device State Information
  - Distributed Detection

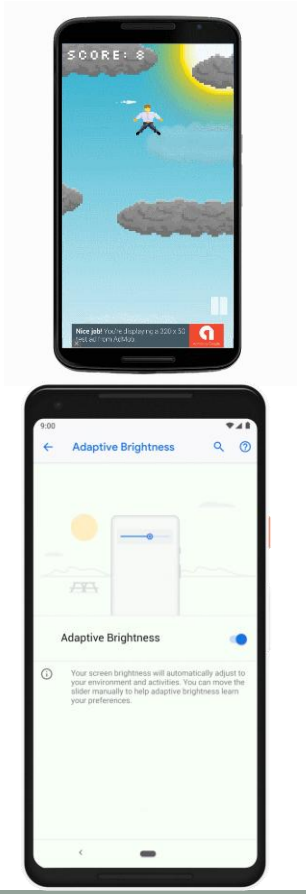
Unique Device Information

- **Basic**
  - Unrealistic/null value for IMEI, Phone No. etc.
- **Smart**
  - Realistic but fixed values

	IMEI	Phone No.	ICCID
	123456789012347	90139442364	89914105611117910720
	null/00000000000	15555215554	89014103211118510720
cuckoo DROiD	351451208401216	97259916243	89963040082067415160
cuckoo DROiD	351451208401216	97259916243	89963040082067415160

Sensors

- Different sensors in a smart phone
  - Motion Sensors: accelerometer, gyroscope
  - Environmental Sensors: illumination (light), humidity
- Detection:
  - Reading: No change in sensors reading



GPS Information

- No change in GPS location
- Use of mock location API to provide fake location
- No correlation with BTS geo-location

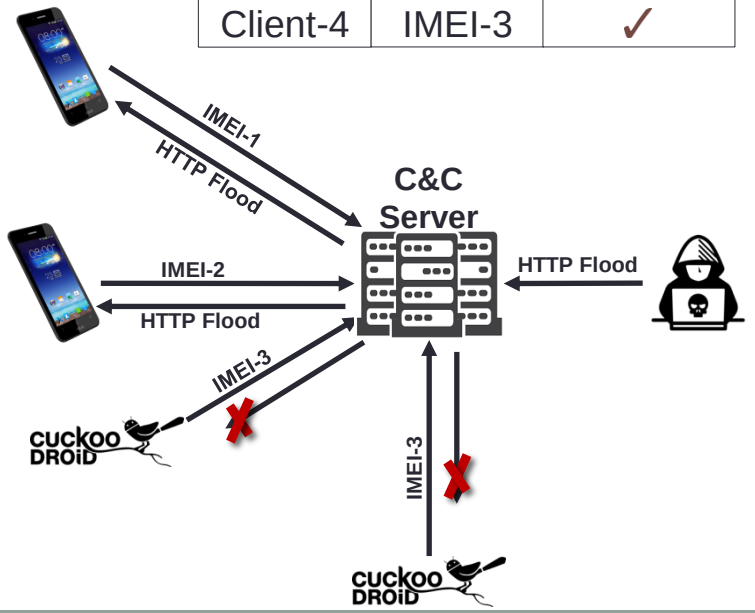
Device State Information

- Smartphone state may change due to:
  - Battery power
  - Signal Strength
  - SMS
  - Call
- No device state change behavior in emulated platform

Distributed Detection

- Detection on server
  - App communicates with server
  - Observing identical information for multiple device like IMEI
- Example:
  - Botnet

Client No.	IMEI	Emulated?
Client-1	IMEI-1	✗
Client-2	IMEI-2	✗
Client-3	IMEI-3	✗
Client-4	IMEI-3	✓



Existing Frameworks Evaluation

Detection Type	Emulator	DroidBox	CuckooDroid	MobSF
Unique ID (Basic)	✓	✗	✗	✗
Unique ID (Smart)	✓	✓	✓	✓
Sensors reading	✓	✓	✓	✓
Device State	✓	✓	✓	✓
GPS	✓	✓	✓	✓
Distributed Detection	✓	✓	✓	✓

Every framework fails to defend against all the detection method except for basic unique ID

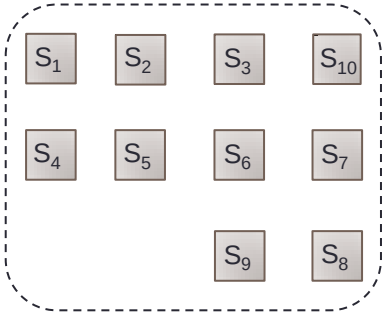
Observation: Emulation-Detection

- Existing framework fails to
  - Generate realistic sensors data (Sensors and GPS)
  - Provide unique identity for telephony system
  - Simulate device state change behavior
- Need a robust anti-emulation-detection system:
  - Hide underline emulated platform
  - Remain undetected when attack is performed from any layer

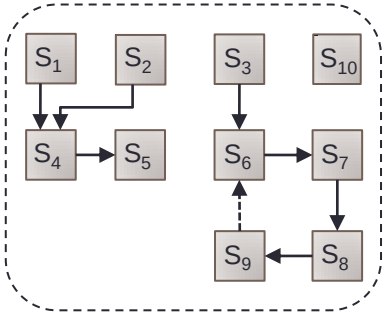
Main challenge lies in generating realistic data for sensors

Realistic Sensors Data Generation

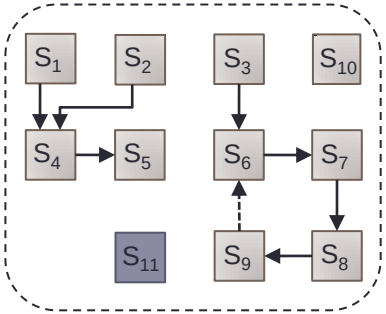
Challenges: Three challenges to generate realistic data for sensors



Sensors value should fluctuate with respect to time



Detection of emulation environment through sensor correlation



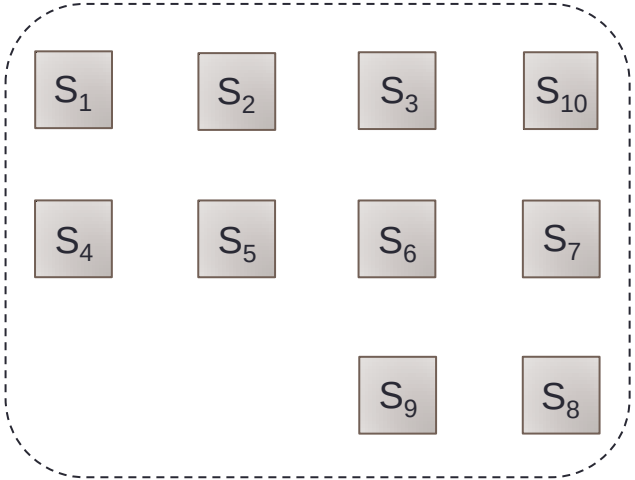
Flexible to incorporate new sensors and sensor relations.

Sensor Data Generation

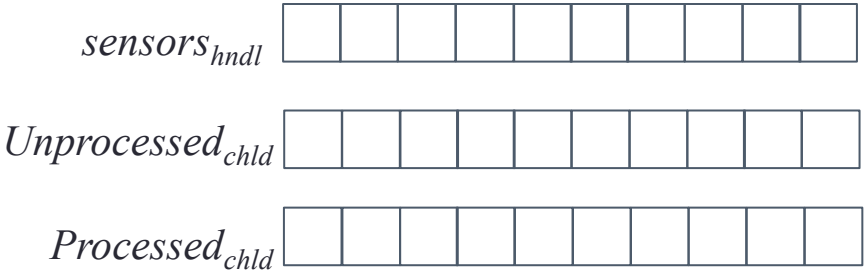
- Input: Requirement from user
  - List of available sensor along with default value generator
  - List of dependency between sensors
    - $S_i \rightarrow S_j$
- Output:
  - Ordered list of sensor handler to generate sensors data

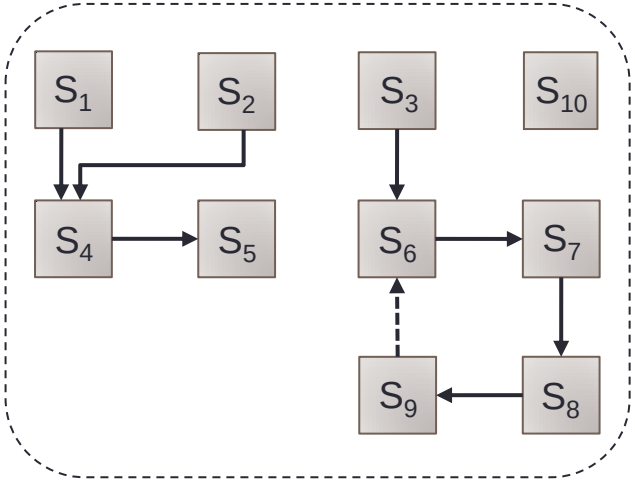
Challenge 1: Independent Sensor

- No dependency between sensors
- Use default value generator for a sensors

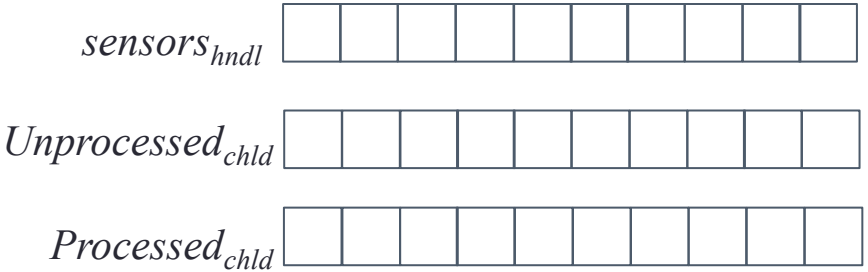


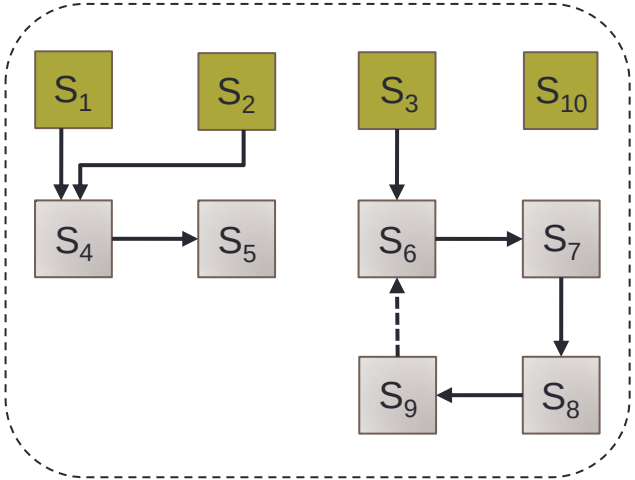
Challenge 2: Dependent Sensor



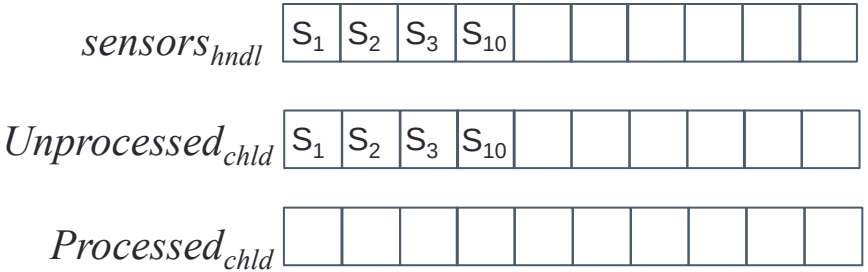


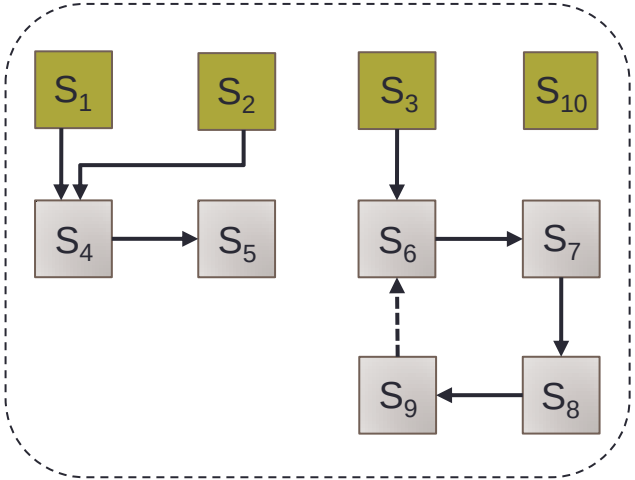
Challenge 2: Dependent Sensor….





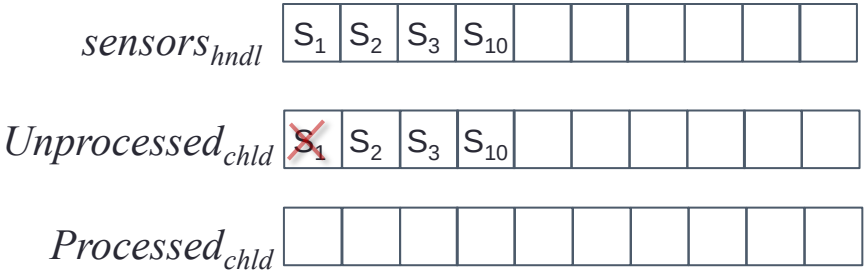
Challenge 2: Dependent Sensor….

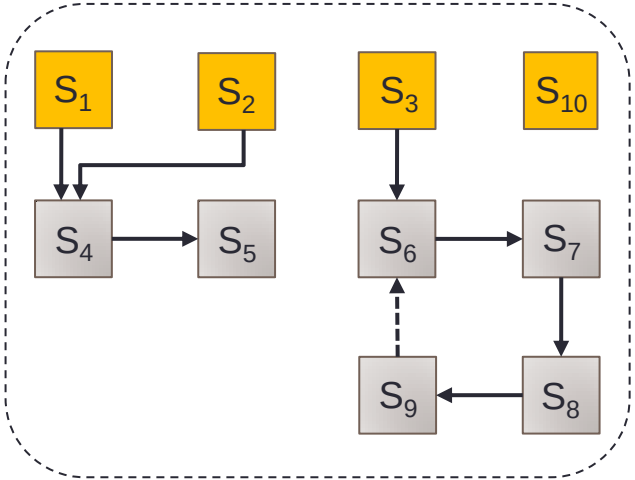




Generate handle for dependent sensors until $Unprocessed_{chld}$ queue is empty

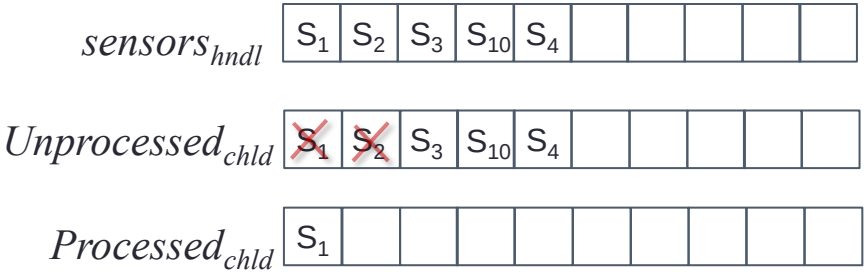
Challenge 2: Dependent Sensor….

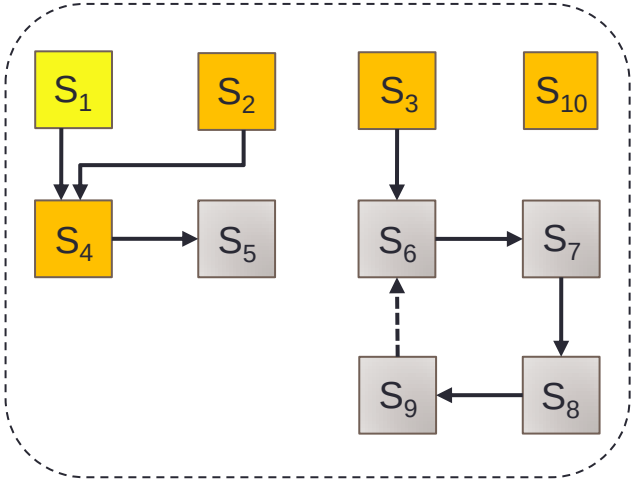




If dependent sensor not in $sensor_{hndl}$, generate handle and add in $sensor_{hndle}$

Challenge 2: Dependent Sensor….

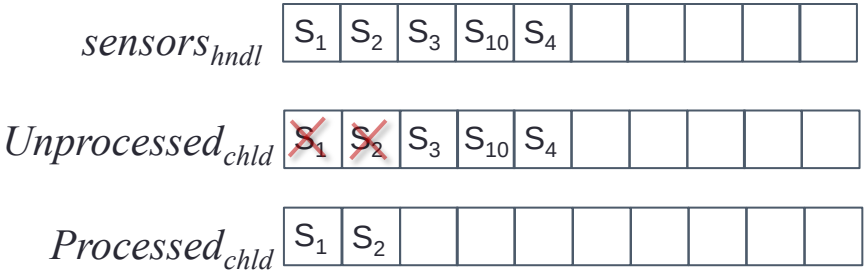


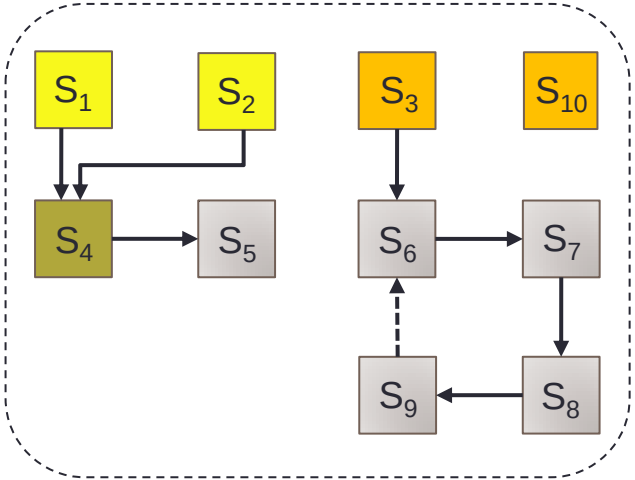


If dependent sensor not in $sensor_{hndl}$, generate handle and add in $sensor_{hndle}$

If dependent sensor not in $Unprocessed_{chld}$, add it

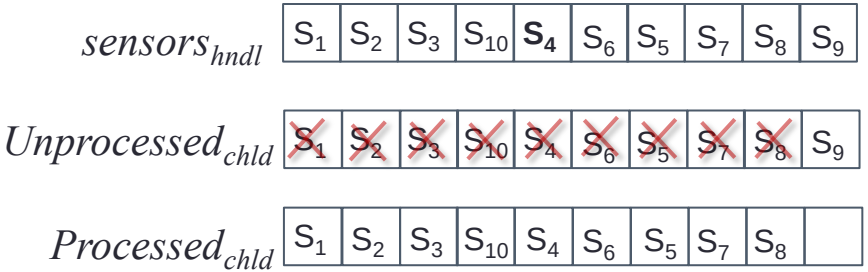
Challenge 2: Dependent Sensor….

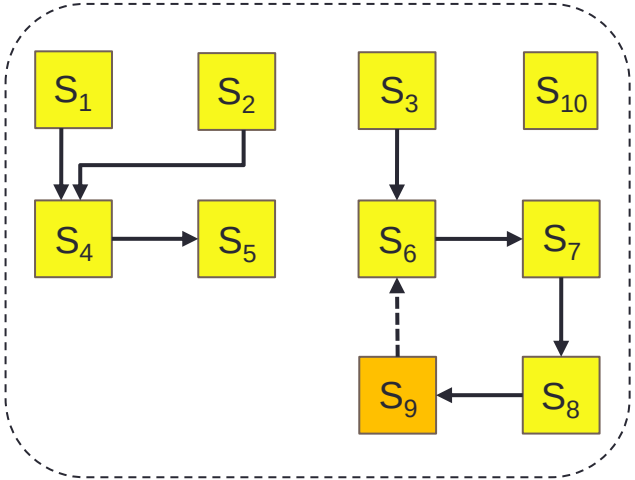




If dependent sensor is in $sensor_{hndl}$, update sensor handle

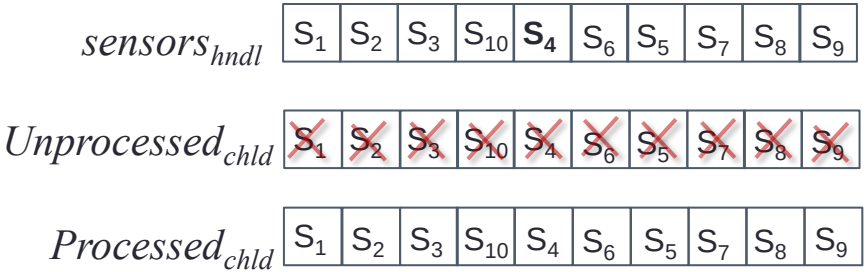
Challenge 2: Dependent Sensor….

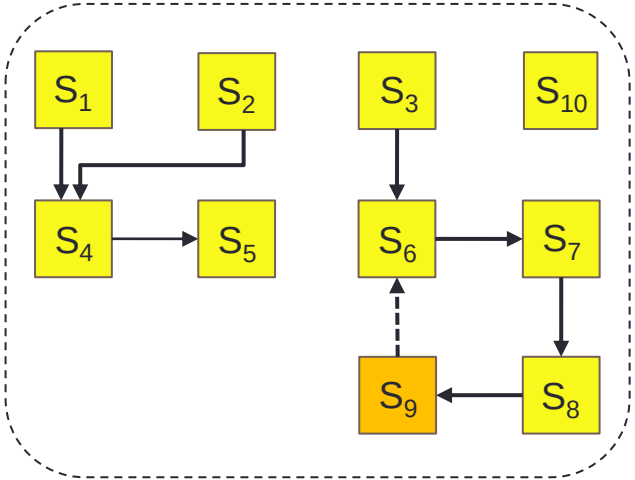




Issue: Cyclic dependency between sensors

Challenge 2: Dependent Sensor….



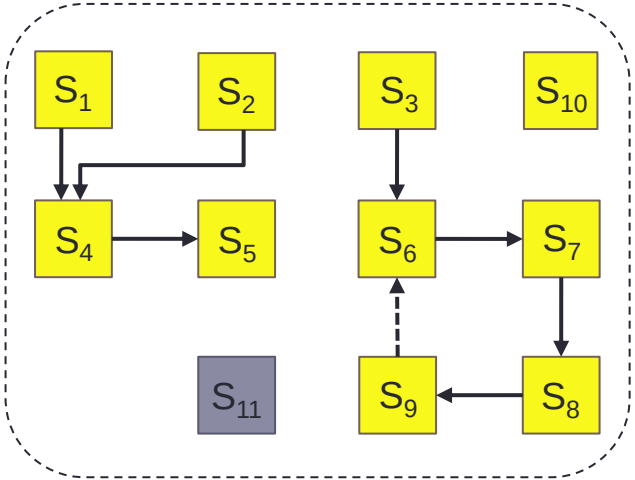


If dependent sensor is in $Processed_{chld}$ list, update handle based on old value of parent sensor

Do not add dependent sensor in $Unprocessed_{chld}$

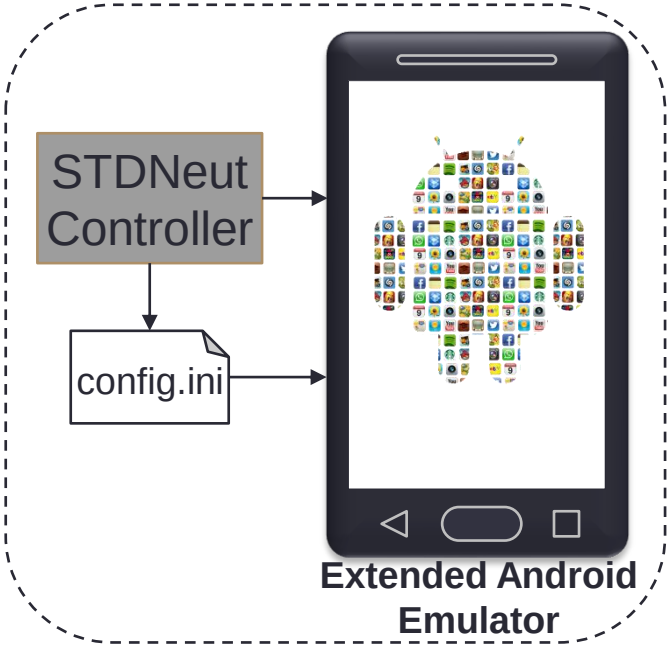
Challenge 3: Adding New Sensor

- Automatically handled if added in:
  - Available sensors list
  - List of dependent sensor



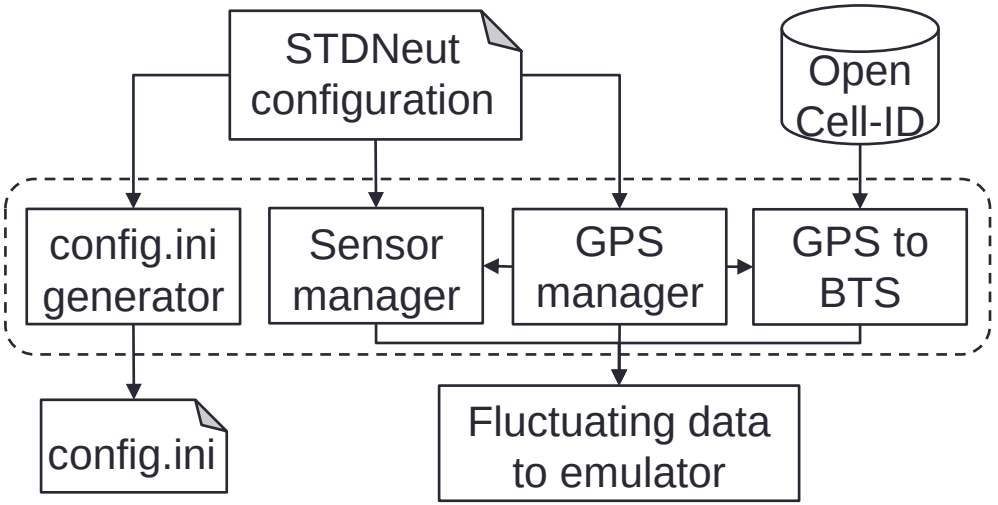
STDNeut Overview

- Two main core component
- STDNeut Controller
  - Launch an Application inside emulator
  - Feeding essential information for anti-emulation-detection like sensors data
- Extended Android Emulator
  - Spoof information for sensors, telephony system and device state



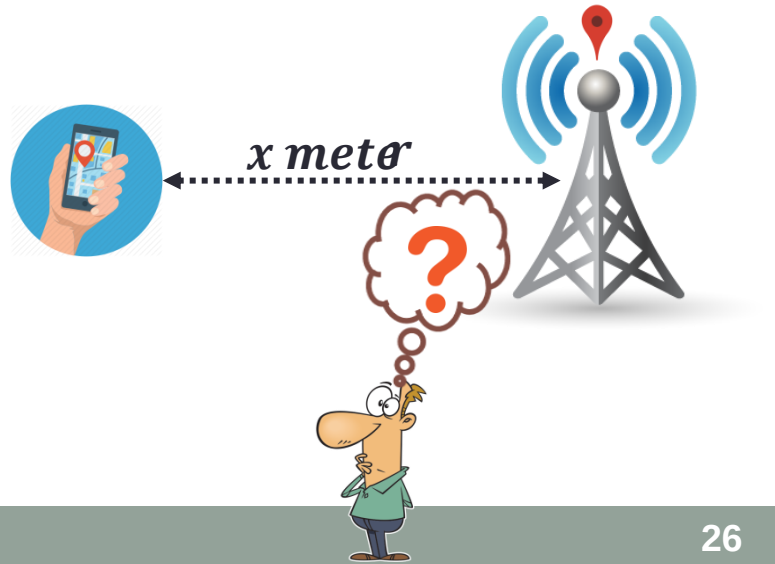
STDNeut Controller

- Four core component
- Sensor data generation is used for sensors values
- Why Separate manager for GPS?



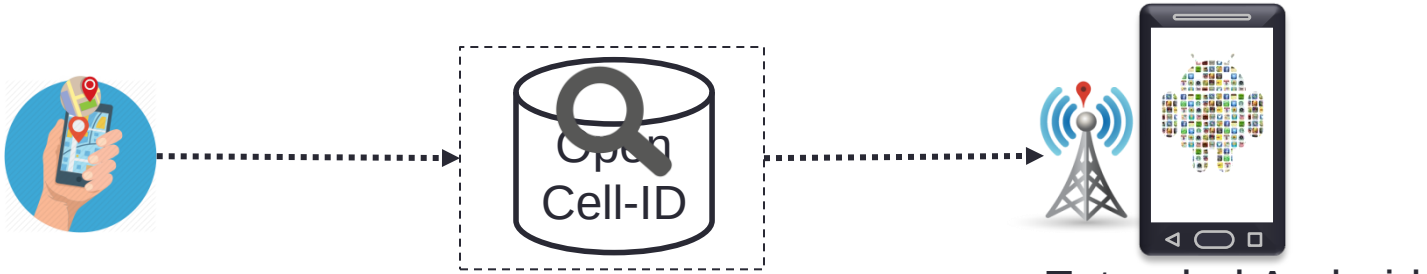
GPS Manager

- Correlation between current location and previous location
- Example:
  - A person cannot reach New York from Washington in 5 min
- Solution: Use path patching algorithm to obtain route
- Other Issue:
  - Correlation with Cell Tower location



GPS to BTS

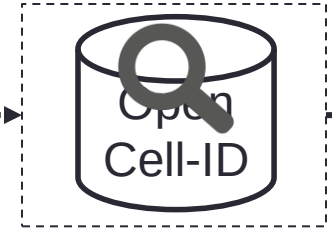
- Get current GPS location from GPS manager
- Obtain nearest Cell Tower ID
  - Uses Open Cell-ID database
  - SIM information
- Feed data to Extended Android Emulator



GPS to BTS

- Get current GPS location from GPS manager
- Obtain nearest Cell Tower ID
  - Uses Open Cell-ID database
  - SIM information
- Feed data to Extended Android Emulator

GPS Manager

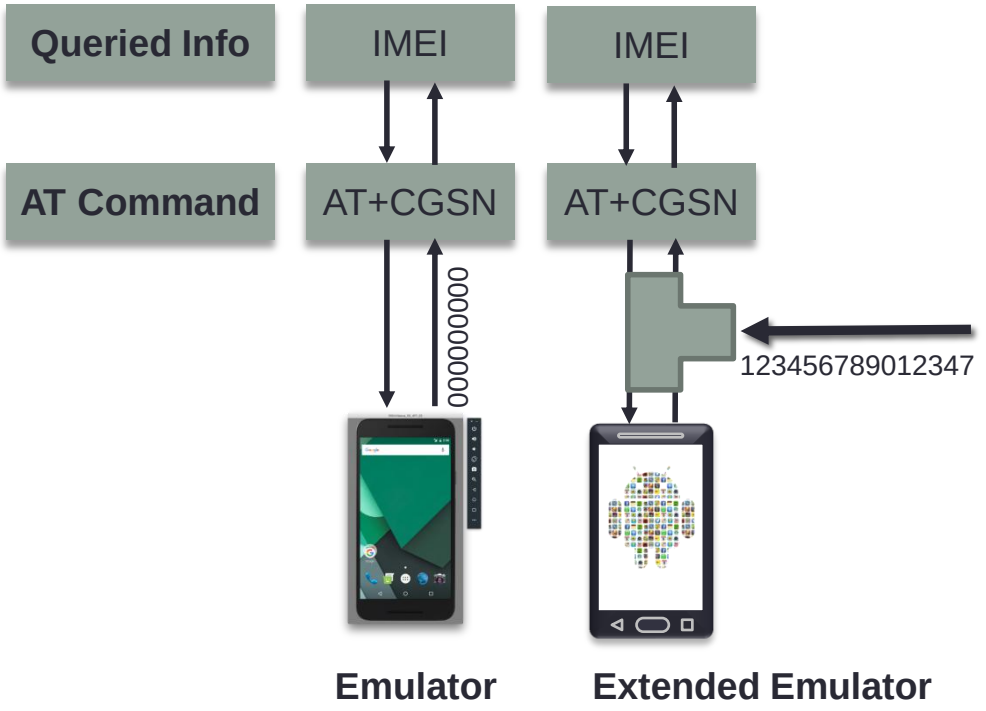


GPS to BTS

Extended Android Emulator

Extended Android Emulator

- Spoof telephony information
  - AT Commands are used to get telephony information
  - Intercept AT commands at Qemu layer:
    - To spoof IMEI, IMSI etc.
  - Change Cell tower ID:
    - Interface with emulator console

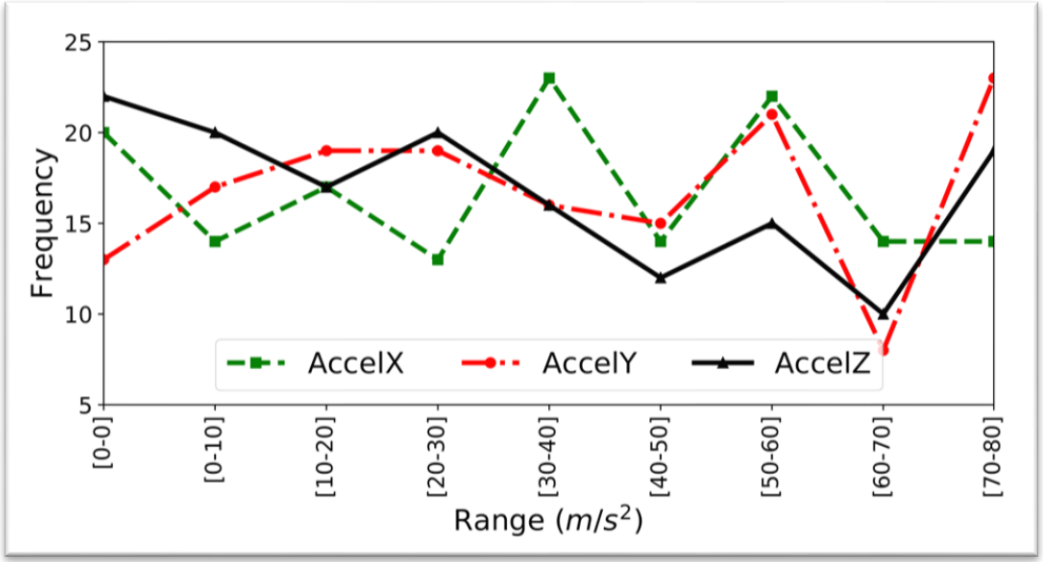


STDNeut Evaluation

- AVD instance is configured with
  - Uses AOSP 7.1
  - Two CPU core
  - 1.5 GB of RAM
  - 2GB internal storage
  - 512MB SD card
  - All sensors
- Effectively hide emulated platform against detection method of emulation-detection library…

Non-detectability through Sensors

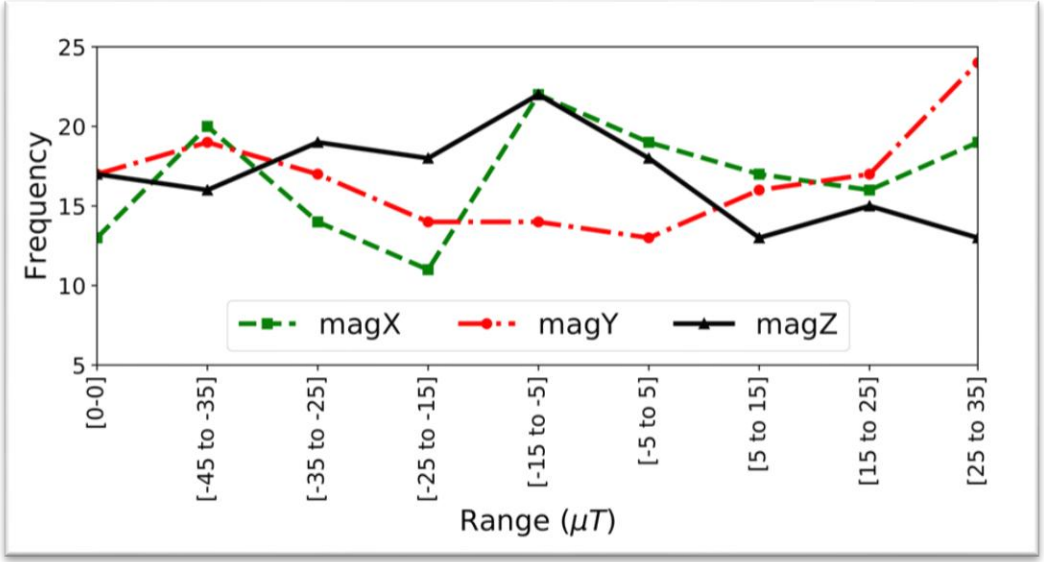
- Developed App to record sensors values
- Uniform distribution of sensors reading
  - Accelerometer



Distribution of accelerometer reading (150 sec)

Non-detectability through Sensors

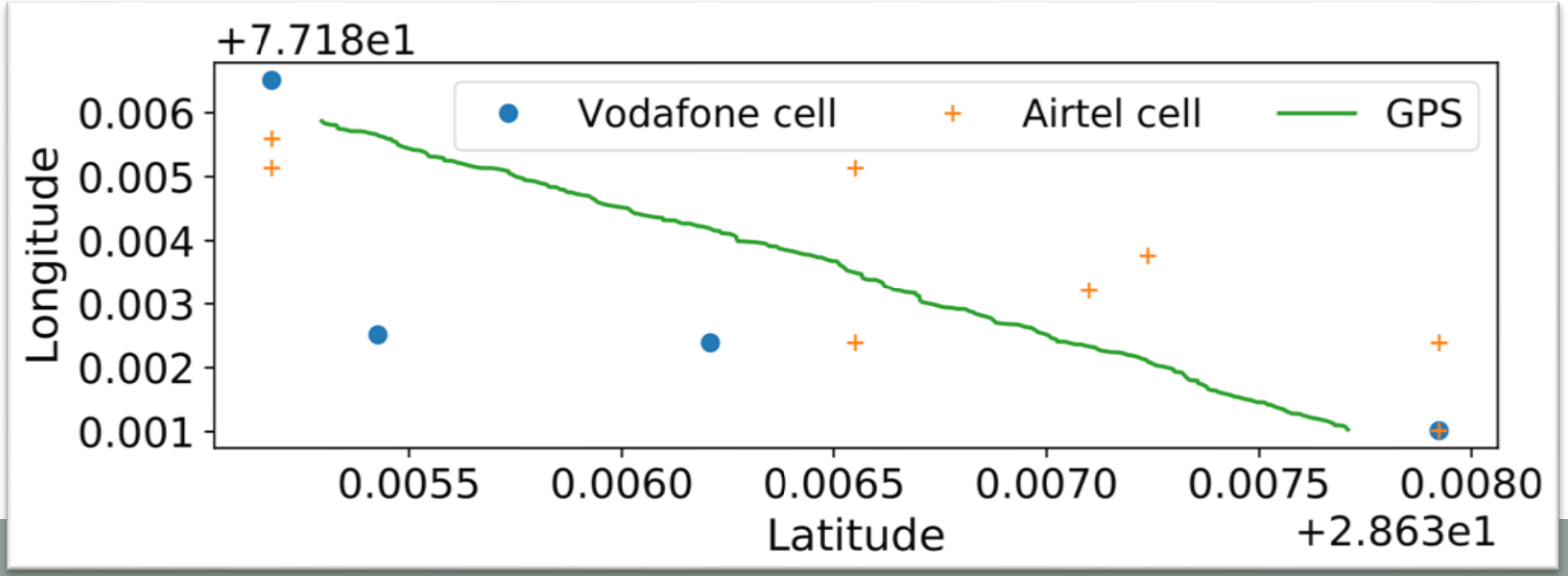
- Developed App to record sensors values
- Uniform distribution of sensors reading
  - Accelerometer
  - Magnetometer



Distribution of magnetometer reading (150 sec)

Non-detectability through Sensors…

- For GPS:
  - Route between (28.63771,77.18103) and (28.63529,77.18586)
  - Cell Tower location with Vodafone and Airtel ($x = 50^0\ mete$)



Non-detectability through Device Info

- Created three instances of STDNeut
- Logged device related unique information using SimCardInfo App:
  - IMEI, IMSI, PhoneNumber, etc.

Queried Information	Information retrieved		
	AVD-1	AVD-2	AVD3
IMEI	359470010002931	359470010302943	359470010002949
IMSI	405541385237906	405521385237806	405511385238906
PhoneNumber	9876543210	9856543410	9876573213

Non-detectability through Device Info

- Created three instances of STDNeut
- Logged device related unique information using SimCardInfo App:
  - IMEI, IMSI, PhoneNumber, etc.

Queried Information	Information retrieved		
	AVD-1	AVD-2	AVD3
IMEI	359470010002931	359470010302943	359470010002949
IMSI	405541385237906	405521385237806	405511385238906

Providing unique information to each virtual device

Possible Improvements to STDNeut

- STDNeut aims to neutralize sensors, telephony system and device state information
- Detection can be possible through:
  - Timing channel attack
  - Qemu specific file information and system properties

Conclusion

EmuDetLib: A flexible emulation-detection library

Existing analysis framework fails to hide emulated platform

Designed STDNeut by insights learn from the evaluation of existing frameworks

STDNeut effectively neutralized the sensros, telephony system and device state information for emulated platform

Thank You