TU Graz

SCIENCE
PASSION
TECHNOLOGY

AndroGUARD: Mitigation of Sensor Fingerprinting on Android

Gergö Kranz

20.02.2025

Outline

1 Introduction
2 Background
3 Sensor Fingerprinting
4 Methodology
5 Approach
6 Implementation
7 Evaluation
8 Discussion & Limitations

Introduction

- Misuse of the Android API
- Used for targeted advertisements
- Does not require user permission

Introduction

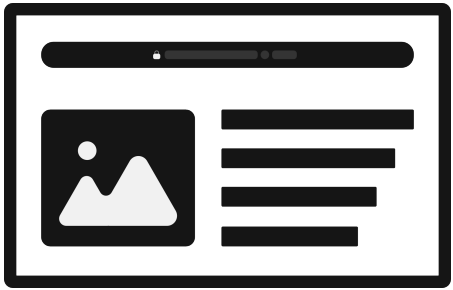
- Misuse of the Android API
- Used for targeted advertisements
- Does not require user permission

Introduction

- Misuse of the Android API
- Used for targeted advertisements
- Does not require user permission

Browser Fingerprinting Methodologies

- Analyzing various browser-specific attributes
- Can be used to distinguish users across sessions



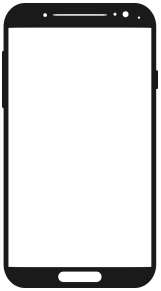
Browser Fingerprinting Protections

- Blocking the execution of JavaScript
- Introduction of controlled randomization

Figure: JShelter

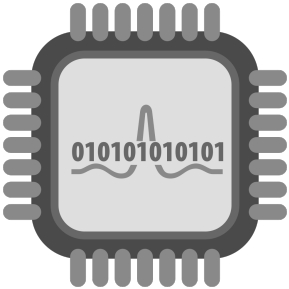
Smartphone Fingerprinting

- Zero permission identifiers
- Personalized configurations



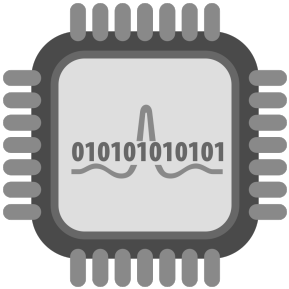
Fingerprinting Sensors

- Measurement inaccuracy of sensors
- Simple to fingerprint via machine learning algorithmus
- Constant over the sensors lifetime



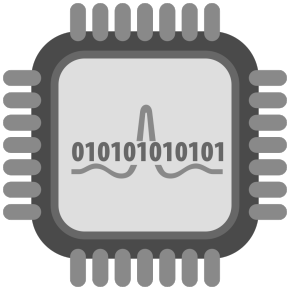
Fingerprinting Sensors

- Measurement inaccuracy of sensors
- Simple to fingerprint via machine learning algorithmus
- Constant over the sensors lifetime



Fingerprinting Sensors

- Measurement inaccuracy of sensors
- Simple to fingerprint via machine learning algorithmus
- Constant over the sensors lifetime

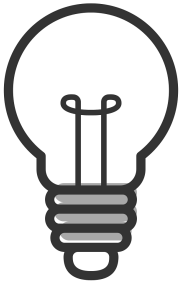


Main Question

How to protect against sensor fingerprinting

Proposed Solutions

- Calibration
- Noise Generation



Calibration

- Systematic adjustment of sensor readings
- Correcting the sensor data

Proposed Solutions

- Calibration
- Noise Generation

Noise Generation

- Introduces variability into the sensor data
- Masks the original values

Challenges

- Calibration
- Noise Generation

Calibration

- Requires user awareness and interaction
- Requires precision

Challenges

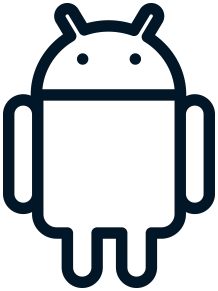
- Calibration
- Noise Generation

Noise Generation

- Degrade the functionality of applications
- Code has to be modified

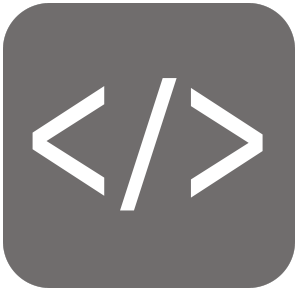
Our Methodology

- Noise Generation
- Patch application vie A2P2 framework



Modifying the Sensor API

- Intercept calls to registerListener method
- Provide modified values to onSensorChanged method



Noise Generation

- Adds random gain and offset to every value
- Masks values
- Loss of precision

Implementation

- Intercept Method
- Noise Generating Function
- Random Value Generation Function

Intercept Method

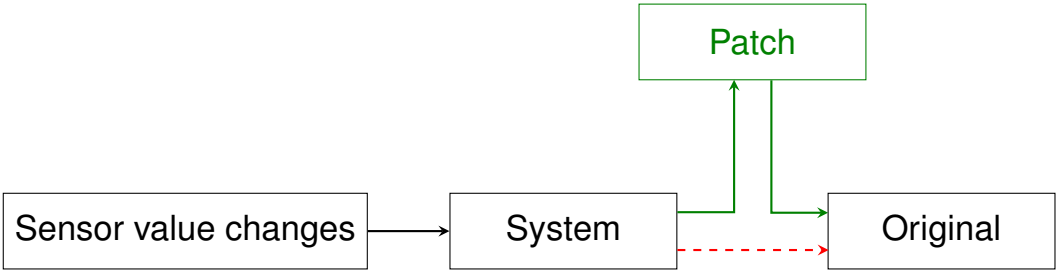


Figure: The function calls from the system are intercepted by our patch and forwarded after modification to the original function.

Implementation

- Intercept Method
- Noise Generating Function
- Random Value Generation Function

Implementation

- Intercept Method
- Noise Generating Function
- Random Value Generation Function

Application of Patch

- Intercept the original method
- Apply appropriate random noise
- Return obstructed sensor data to original method

Application of Patch

- Intercept the original method
- Apply appropriate random noise
- Return obstructed sensor data to original method

Application of Patch

- Intercept the original method
- Apply appropriate random noise
- Return obstructed sensor data to original method

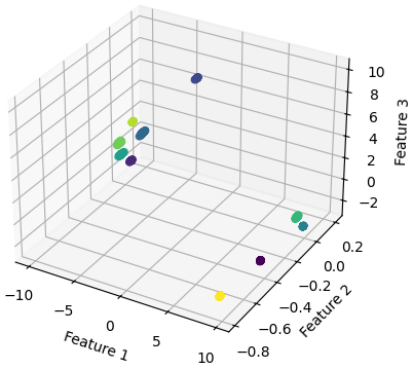
Testing

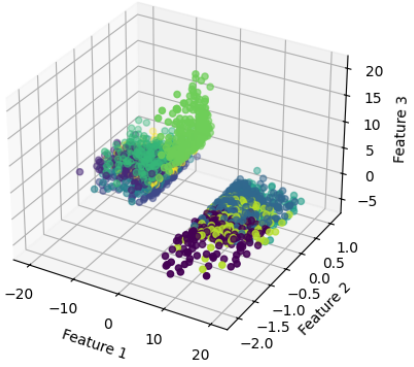
- Functionality
- Effectiveness
- Usabilty

Testing

- Functionality
- Effectiveness
- Usabilty

Effectiveness



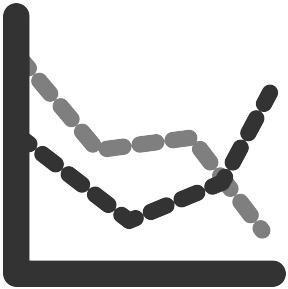


Testing

- Functionality
- Effectiveness
- Usabilty

Noise Level Adjustment

- Increasing noise decreases fingerprintability
- Increasing noise decreases functionality
- Noise has to be balanced between effectiveness and functionality



Discussion & Limitations

- Comparing values before and after the patch
- Could not be done sufficiently due to limited access to supported hardware

Conclusion

- Masking the sensor values decreases fingerprintability
- Modifying the SensorEventListener makes it easy to incorporate the patch into the Android API