Using Taint Tracking to Improve Energy Efficiency of Always-on Smartphone Apps

Haichen Shen*, Aruna Balasubramanian*, Anthony LaMarca†, David Wetherall*
*{haichen, arunab, djw}@cs.washington.edu  †anthony.lamarca@intel.com

Motivation
• Sensor hubs, a low cost always-on micro-controller, can collect sensor data without waking the CPU
  BUT without understanding sensor usage patterns in real apps, it’s difficult to design a sensor hub architecture

Goal
• Taint Tacking: Understand sensor usage patterns in continuous monitoring applications

Approach: Taint Tracking

What to track
• When are sensor data collected?
• When are they needed for update?
• How often is collected data used?

Why not static analysis?
• Difficult to analyze data and control flow in complex apps
• Runtime tracking gives us the sensor usage in real world

Challenge
• Taint tracking, used typically for privacy, only tracks data flow
• Data flow tracking alone is not sufficient to track sensor data

Example: Pedometer App

```java
void onSensorChange(sensorEvent)
{
    val = sensorEvent.val;
    direct = signal(val);
    blockTag = getTag(val) | getTag(direct) | getTag(lastDirect);
    if (val > THRESHOLD &
        direct != lastDirect)
    {
        stepCounter++;
        Taint(stepCounter, blockTag);
        lastDirect = direct;
        Taint(lastDirect, blockTag);
    }
    // every 10 seconds
    updateUI(stepCounter);
}
```

System Overview

**Product:**
- Sensor reading in Android runtime
- Add Taint tag & index to sensor data
- App get sensor data & data flow taint
- Get the block tag from variable in the condition
- Use Block tag to taint variables inside this block
- User-perceivable action (update UI, send to network, etc)

Example: Pedometer App

```java
void onSensorChange(sensorEvent)
{
    val = sensorEvent.val;
    direct = signal(val);
    blockTag = getTag(val) | getTag(direct) | getTag(lastDirect);
    if (val > THRESHOLD &
        direct != lastDirect)
    {
        stepCounter++;
        Taint(stepCounter, blockTag);
        lastDirect = direct;
        Taint(lastDirect, blockTag);
    }
    // every 10 seconds
    updateUI(stepCounter);
}
```

Result

Using our taint tracking tool to analyze the sensor usage of the Pedometer application.

20% of the time, the app only updates once in a user-perceivable manner after 30 contiguous sensor readings.

Future Work

• Automatically instrument the application
• Analyze many popular off-the-shelf sensor applications
• Understand the energy implications of different sensor hub tasks