Sentinel: Generating GUI Tests for Android Sensor Leaks

Haowei Wu, Yan Wang, Atanas (Nasko) Rountev

Ohio State University

Problem

Sensors in Android devices

- Accelerometer, proximity sensor, light sensor, temperature sensor, air pressure sensor, …

Sensor leaks

- "As a best practice you should always disable sensors you don't need" [Google documentation]
- Leak: failure to disable an unnecessary sensor
- Can cause battery drain

Our goal: test generation to uncover sensor leaks

Sentinel: Sensor Testing to Detect Leaks

Patterns of sensor leaks: an app component acquires a sensor but does not release it

Static analysis

- Model of sensor-related effects of GUI events
- "Suspicious" paths in the model

Testing

- Generation/execution of test cases from model paths

Experimental evaluation

- Small number of test cases (even for complex apps)
- Most generated test cases uncover real leaks

Sensor-Related Code

Sensor categories represented by integer constants

- Sensor.TYPE_ACCELEROMETER, Sensor.TYPE_LIGHT, …

Sensor object for each category

- Sensor s = getDefaultSensor(type)

Sensor listener object

- SensorEventListener l = new SensorEventListener() { … }

Acquire/release operations

- registerListener(l,s)
- unregisterListener(l,s)

Example of Sensor Usage

```
class SettingActivity extends Activity {

void onCreate() {
    Button b = ...;
    b.setOnClickListener(this); }

void onClick() {
    Intent i = new Intent(UnlockActivity.class);
    startActivity(i);}}
```

Example of Sensor Usage

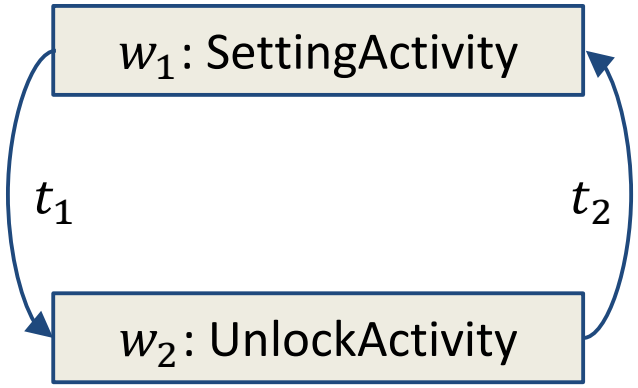
```
class SettingActivity extends Activity {

void onCreate() {
    Button b = ...;
    b.setOnClickListener(this); }

void onClick() {
    Intent i = new Intent(UnlockActivity.class);
    startActivity(i);}}
```

w_1: SettingActivity

w_2: UnlockActivity

Example of Sensor Usage

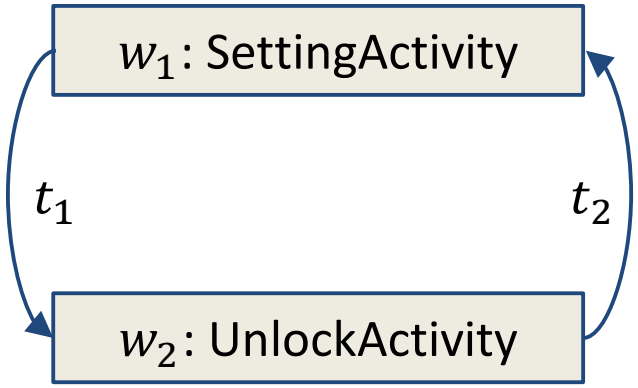
```
class SettingActivity extends Activity {

void onCreate() {
    Button b = ...;
    b.setOnClickListener(this); }

void onClick() {
    Intent i = new Intent(UnlockActivity.class);
    startActivity(i);}}

class UnlockActivity extends Activity {

void onCreate() { … }

void onDestroy() { … }}
```



t_1: click on button b
SettingActivity.onClick
UnlockActivity.onCreate
t_2: BACK
UnlockActivity.onDestroy

Example of Sensor Usage

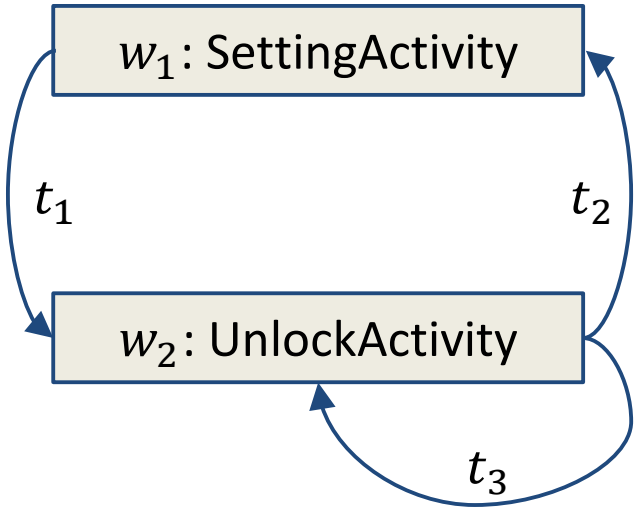
```
class SettingActivity ...

class UnlockActivity extends Activity {

void onCreate() {

    SwitchCompact sc = …;
    Sensor accel =
        getDefaultSensor(TYPE_ACCELEROMETER);
    SensorEventListener lst = …;

    sc.setOnCheckedChangeListener( new
       OnCheckedChangeListener {
          void onCheckedChanged(){
             registerListener(lst, accel);}});}

void onDestroy() { … }}
```



t_1: click on button b
SettingActivity.onClick
UnlockActivity.onCreate
t_2: BACK
UnlockActivity.onDestroy

Example of Sensor Usage

```
class SettingActivity ...

class UnlockActivity extends Activity {

void onCreate() {

    SwitchCompact sc = …;
    Sensor accel =
        getDefaultSensor(TYPE_ACCELEROMETER);
    SensorEventListener lst = …;

    sc.setOnCheckedChangeListener( new
       OnCheckedChangeListener {
          void onCheckedChanged(){
             registerListener(lst, accel);}});}

void onDestroy() { … }}
```



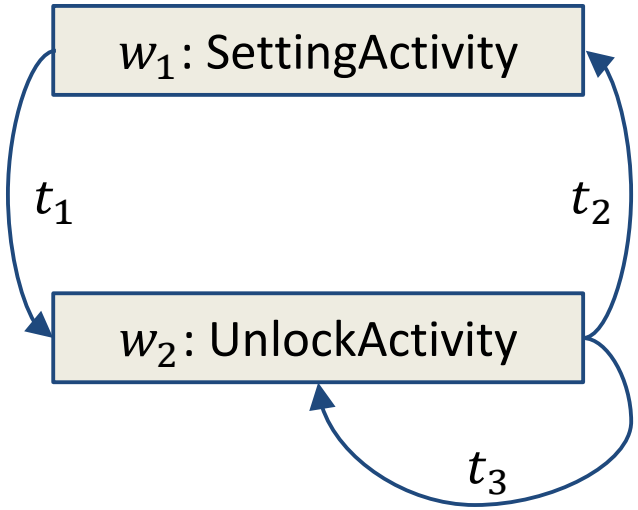
t_3: toggle switch sc

onCheckedChanged

- ✓ register sensor listener
- ✓ wait for a shake gesture from the user

Example of Sensor Usage

```
class SettingActivity …

class UnlockActivity extends Activity {

void onCreate() {
   …
   SensorEventListener lst =
      new SensorEventListener {
        void onSensorChanged() {
          if (…) unregisterListener(lst, accel);}};
   … }

void onDestroy() { … }}
```



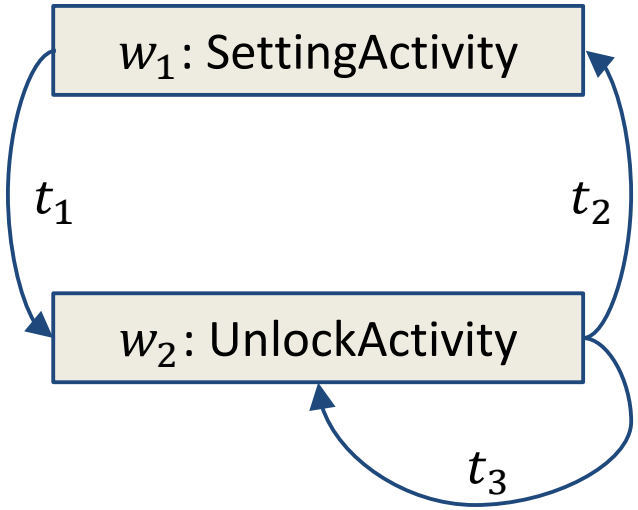
when the shake gesture occurs

onSensorChanged

- ✓ unregister sensor listener if the movement is “large enough” to be a shake

Example of Sensor Usage

```
class SettingActivity ...

class UnlockActivity extends Activity {

void onCreate() {
   ...
   SensorEventListener lst =
      new SensorEventListener {
        void onSensorChanged() {
          if (...) unregisterListener(lst, accel);}};
   ... }

void onDestroy() { ... }}
```



when the shake gesture occurs
onSensorChanged
✓ unregister sensor listener

But if the user presses BACK and then just exits the app, the listener remains active:
sensor leak

Sentinel: Sensor Testing to Detect Leaks

Patterns of sensor leaks: an app component acquires a sensor but does not release it

Static analysis component

- Model of sensor-related effects of GUI events
- "Suspicious" paths in the model

Testing component

- Generation/execution of test cases from model paths

Experimental evaluation

- Small number of test cases (even for complex apps)
- Most generated test cases uncover real leaks

Static Modeling of GUI Control Flow

Window transition graph

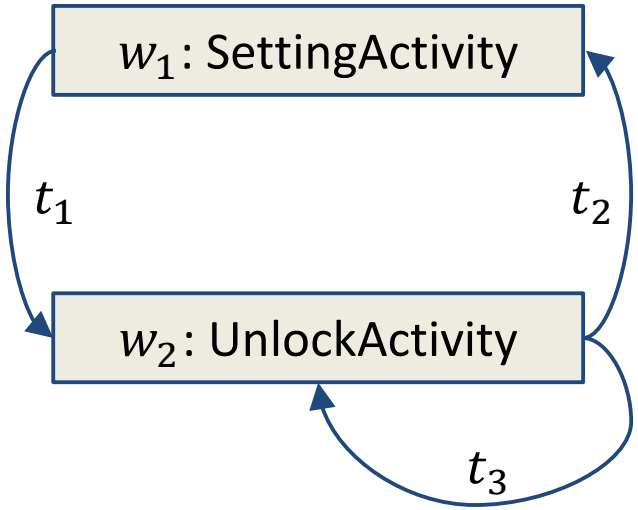
[Yang et al., ASE 2015]

- Windows, widgets, events
- Callbacks respond to events
- Transitions between windows: $open(w)$ and $close(w)$

Sensor effects of callbacks:

$acquire(s)$ and $release(s)$

- s is a pair (listener,sensor)



t_1: click on button b
onClick(b) onCreate(w2)
$open(w_2)$

t_2: BACK
onDestroy(w2)
$close(w_2)$

t_3: toggle switch sc
onCheckedChanged(sc)
$acquire(s)$

Leak Patterns as Context-Free Languages

Alphabet with symbols $open(w)$, $close(w)$, $acquire(s)$, $release(s)$, $suspend(w)$

Lifetime of window w_i: language $L_1(w_i)$

$S \rightarrow open(w_i)\ \ B\ \ close(w_i)$

$B \rightarrow open(w_j)\ \ B\ \ close(w_j) \mid B\ B \mid$

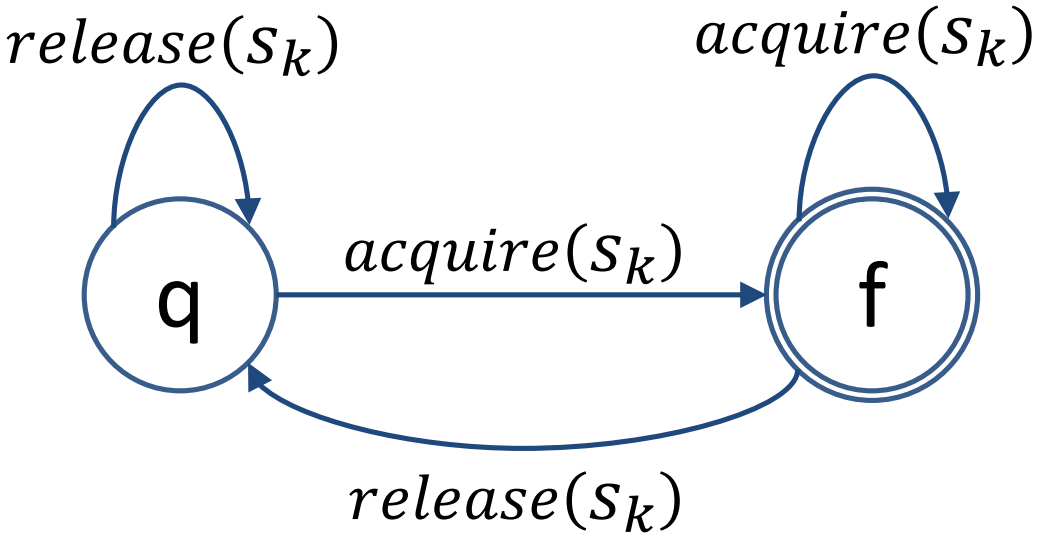
$acquire(s) \mid release\ (s) \mid\ \epsilon$

Suspended state of window w_i: language $L_2(w_i)$

$S \rightarrow open(w_i)\ \ B\ \ suspend(w_i)$

Leak Patterns as Context-Free Languages

Leak of sensor s_k: regular language $R(s_k)$ defined by a finite automaton



Any other alphabet symbol has no effect

Leak Patterns as Context-Free Languages

Leak pattern 1: path with a label string from language $L_1(w_i) \cap R(s_k)$

Window w_i acquires s_k but does not release it by the end of its lifetime

Leak pattern 2: $L_2(w_i) \cap R(s_k)$

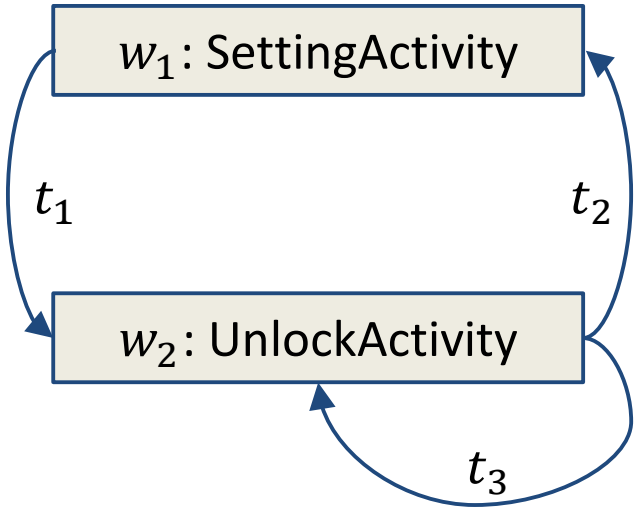
Window w_i acquires s_k and goes to a suspended state without releasing s_k

Running Example

t_1, t_3, t_2: string $open(w_2)$, $acquire(s)$, $close(w_2)$ belongs to $L_1(w_2) \cap R(s)$

This path is a static candidate for a run-time sensor leak

Need to generate and execute a test case to confirm the leak



t_1: click on button b
$open(w_2)$

t_2: BACK
$close(w_2)$

t_3: toggle switch sc
$acquire(s)$

SENTINEL: Sensor Testing to Detect Leaks

Patterns of sensor leaks: an app component acquires a sensor but does not release it

Static analysis component

- Model of sensor-related effects of GUI events
- "Suspicious" paths in the model

Testing component

- Generation/execution of test cases from model paths

Experimental evaluation

- Small number of test cases (even for complex apps)
- Most generated test cases uncover real leaks

Generation of Test Cases

Static path traversal

- Stack of $open(w)$ and $close(w)$ symbols: $close(w)$ is accepted only if the top is a matching $open(w)$
- Track the state of the finite automaton for each s

Reduce number of test cases

- No cycles; limit number of $open(w)$ and $close(w)$
- Only the shortest path from a language
- Only blame the last open w at the time s was acquired

Implementation

- Wrapper for UI Automator
- Android Debug Bridge to find active sensors

Example of a Test Case

```
d.screen.on() # turn on device screen
killApp("com.calculator.vault") # kill app to clean up resources
oldsensors = readSensors() # currently-acquired sensors
startActivity("com.calculator.vault.UnlockActivity")
d(resourceId="com.calculator.vault:id/shake_btn").click() # click the switch widget
d.press.back() # press the back button
newsensors = readSensors() # currently-acquired sensors
# report differences between newsensors and oldsensors
```

Additional setup may be necessary for some test cases: e.g., setting up the password for the vault

Evaluation

Analyzed apps

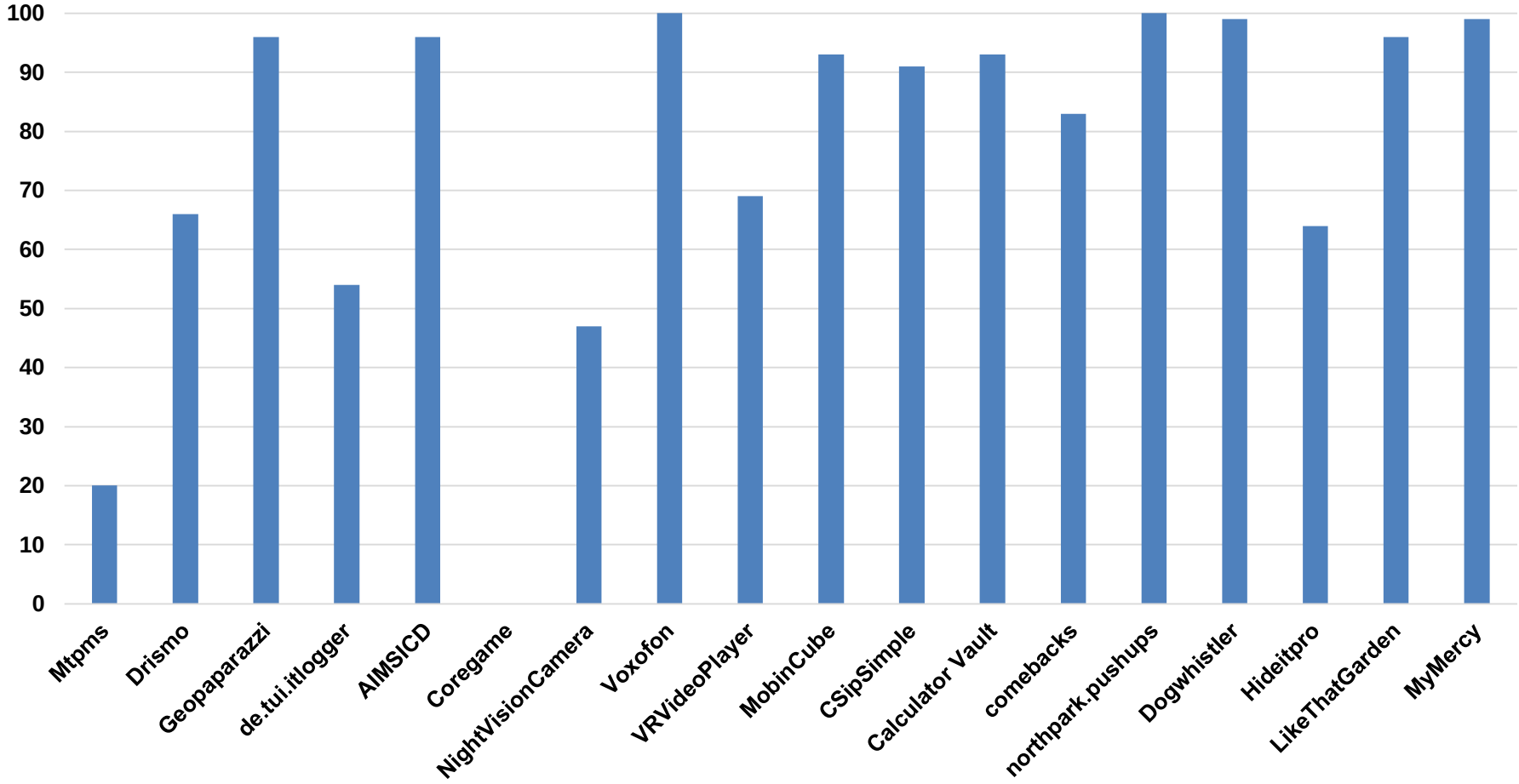
- 709 apps that contain sensor APIs
- 18 apps had instances of the leak patterns
- Cost of static analysis and test generation is practical

36 test cases generated and executed

- 24 test cases exposed run-time leaks
- 9 test cases in 3 apps did not produce leaks due to an overly conservative static analysis
- 3 test cases could not be executed

Conclusion: small number of focused, mostly-effective test cases, even for complex apps

% Reduction in # Paths: L vs $(L \cap R)$



Conclusion: using sensor-aware static analysis dramatically reduced the number of paths

Sentinel: Sensor Testing to Detect Leaks

Two patterns of sensor leaks

Static analysis of

- GUI control flow
- Sensor effects of callbacks along this flow

Static path traversal for test generation

- Generates a small number of test cases
- Most test cases reveal real defects

Static Analysis Details

Input: an APK file

- **Soot** to create a program representation
- **Gator** to create the window transition graph

Sensor effects

- Each callback: onCreate, onClick, onCheckedChanged, …
- $acquire(s)$: registerListener(s) that is not inter-procedurally post-dominated by unregisterListener(s)
- $release(s)$: every interprecedural path contains unregisterListener(s)
- Also analyze callback onSensorChanged