Soteria: Automated IoT Safety and Security Analysis

Z. Berkay Celik
Patrick McDaniel
Gang Tan

Penn State University
USENIX ATC 2018
Internet of Things (IoT) enables the future

Smart Homes

Smart Energy

Healthcare

Smart Farms
IoT is not magic

Connected devices

Mobile app

Automation

IoT application
IoT enables the future (and a whole lot of problems)

Smart home apocalypse
February 27, 2018

Imagine the life smart home developers want you to see:
Your busy day at work is over, and you’re almost home.

A few seconds later, the smart alarm goes nuts, blaring its intruder alert. It was supposed to detect your smartphone’s presence and stand down! At least something seems to be working: The TV is on already — but it is showing a real-time feed of you from the smart camera on the ceiling. And you can hear the sirens of approaching fire engines.

Denning et al., Ronen et al., Fernandes et al., Celik et al.
In this talk...

How do we **ensure IoT implementations and environments adhere to safety and security properties?**

*Soteria* *

* Greek goddess protecting from harm
How safety/security violations happen?

**Individual app**

<table>
<thead>
<tr>
<th>Expected behavior</th>
<th>Actual behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_0$: alarm-off</td>
<td>$S_0$: alarm-off</td>
</tr>
<tr>
<td>smoke</td>
<td>smoke</td>
</tr>
<tr>
<td>$S_1$: alarm-on</td>
<td>$S_1$: alarm-on</td>
</tr>
</tbody>
</table>

Does alarm sound when there is smoke?

**IoT environment**

<table>
<thead>
<tr>
<th>Smoke-Alarm</th>
<th>Water-Leak-Detector</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_0$: alarm-off</td>
<td>$S_0$: water valve-open</td>
</tr>
<tr>
<td>smoke</td>
<td>leak</td>
</tr>
<tr>
<td>$S_1$: alarm-on and water valve-open</td>
<td></td>
</tr>
<tr>
<td>heat $&gt;135^\circ$F</td>
<td>$S_1$: water valve-closed</td>
</tr>
<tr>
<td>Smoke-Alarm</td>
<td>Water-Leak-Detector</td>
</tr>
</tbody>
</table>

Does the sprinkler system active when there is a fire?
Soteria

Problem: IoT platforms cannot evaluate whether an IoT app or environment (collection of apps) is safe, secure, and operates correctly

- Soteria is a **static analysis system** that provides formal verification by **model checking** of IoT apps
State-model extraction from source code

• What is state model?
  ‣ States and transitions
  ‣ In IoT applications...
    - **States**: Device attributes
    - **Transitions**: Events changing the attributes

• Challenges...
  ‣ State-explosion problem
  ‣ Conditional device attribute changes

State-model of an example app
State reduction

- **Property Abstraction:** Reduce states by aggregating numerical-valued attributes.

```python
def modeChangeHandler(evt):
    def temp = 68
    setTemp(temp)

def setTemp(t):
    ther.setHeatingPoint(t)
```

Without property abstraction

- Thermostat temperature

- Worklist
  - t=50
  - t=51
  - t=95

With property abstraction

- Thermostat temperature

- Worklist
  - (2: temp = 68)
  - (6: t, 3: temp)
  - (6: t)

Soteria prunes infeasible paths using path- and context-sensitivity.
Conditional device attribute changes

- Perform path exploration and accumulate path conditions
  - Add a transition using end states and path conditions

```java
subscribe(presence, present, handler)
// Entry point
handler()
{
    above = 50
    below = 5
    power_val = get_power()
    if(power_val > above){
        switch.off()
    }
    if(power_val < below){
        switch.on()
    }
}
```

```java
get_power(){
    latest_pow=power_meter.currentValue("power")
    return latest_pow
}
```

Without path exploration
- Present
  - Power > 50
  - Power < 5

With path exploration
- Present
  - Power < 5
IoT safety/security property identification

• **Property** is a system artifact that formally expressed via temporal logic and validated on the state-model

• **General properties**
  - Independent of app’s semantics

• **App-specific properties**
  - Identifies use cases of one or more devices

1. Attributes of conflicting values
2. Independent of app’s semantics
3. Race condition of events

- The door must always be locked when the user is not home
- The refrigerator and security system must always be on
- The water valve must be closed if a leak is detected
- The alarm must always go off when there is smoke
Property validation

• Individual apps
  ‣ **General properties are** verified at state-model extraction time
  ‣ **App-specific properties** are validated through a model checker

• Multi-apps
  ‣ Apps often interact through a common device
    ‣ Create a **union state-model** of interacting apps

<table>
<thead>
<tr>
<th></th>
<th>switch-off</th>
<th>smoke-detected</th>
<th>switch-on</th>
</tr>
</thead>
<tbody>
<tr>
<td>App1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>App2</td>
<td>away-mode</td>
<td>switch-on</td>
<td>home-mode</td>
</tr>
<tr>
<td>App3</td>
<td>door-unlocked</td>
<td>home-mode</td>
<td>door-locked</td>
</tr>
</tbody>
</table>
Evaluation

- Implemented Soteria for SmartThings IoT platform
- Selected **65** SmartThings market apps with bias on popularity and access to various devices

<table>
<thead>
<tr>
<th>Apps*</th>
<th>Nr.</th>
<th>Unique Devices</th>
<th>Avg/Max States</th>
</tr>
</thead>
<tbody>
<tr>
<td>Official</td>
<td>35</td>
<td>14</td>
<td>36/180</td>
</tr>
<tr>
<td>Third-party</td>
<td>30</td>
<td>18</td>
<td>32/96</td>
</tr>
</tbody>
</table>

*App functionality: Safety and security, green living, convenience, home automation, and personal care*
## Findings - Individual app analysis

- **Nine (14%)** individual apps violate **10 (29%)** properties

<table>
<thead>
<tr>
<th>App ID</th>
<th>Violation Description</th>
<th>Violated Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>TP1</td>
<td>The music player is turned on when user is not at home</td>
<td>P.13</td>
</tr>
<tr>
<td>TP2</td>
<td>The door is unlocked on sunrise and locked on sunset</td>
<td>P.1</td>
</tr>
<tr>
<td>TP3</td>
<td>The location is changed to the different modes when the switch is turned off and when the motion is inactive</td>
<td>S.4</td>
</tr>
<tr>
<td>TP4</td>
<td>The flood sensor sounds alarm when there is no water</td>
<td>P.29</td>
</tr>
<tr>
<td>TP5</td>
<td>The music player turns on when the user is sleeping</td>
<td>P.28</td>
</tr>
<tr>
<td>TP6</td>
<td>The lights turn on and turn off when nobody is at home</td>
<td>P.13, S.1</td>
</tr>
<tr>
<td>TP7</td>
<td>The lights turn on and turn off when the icon of the app is tapped</td>
<td>S.1</td>
</tr>
<tr>
<td>TP8</td>
<td>The switch turns on and blinks lights when no user is present</td>
<td>P.12</td>
</tr>
<tr>
<td>TP9</td>
<td>The door is locked multiple times after it is closed</td>
<td>S.2</td>
</tr>
</tbody>
</table>

TP = Third-party  
S = General properties  
P = App-specific properties
Findings - Multi-app analysis

- **17 (26%)** apps interacting in **three** groups and violate **11 (31%)** properties

<table>
<thead>
<tr>
<th>Gr. ID</th>
<th>App ID</th>
<th>Event/Actions</th>
<th>Violated Pr.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>O3</td>
<td>contact sensor open</td>
<td>switch-on</td>
</tr>
<tr>
<td></td>
<td>O4</td>
<td>contact sensor open</td>
<td>switch-off</td>
</tr>
<tr>
<td></td>
<td></td>
<td>contact sensor close</td>
<td>switch-off</td>
</tr>
<tr>
<td></td>
<td>O8, TP12</td>
<td>contact sensor open</td>
<td>switch-off</td>
</tr>
<tr>
<td>2</td>
<td>O14</td>
<td>contact sensor open</td>
<td>switch-off</td>
</tr>
<tr>
<td></td>
<td>O9, O16, TP3</td>
<td>motion active</td>
<td>switch-on</td>
</tr>
<tr>
<td></td>
<td>TP2</td>
<td>app touch</td>
<td>switch-on</td>
</tr>
<tr>
<td>3</td>
<td>O7, TP3</td>
<td>switch off</td>
<td>change location mode</td>
</tr>
<tr>
<td></td>
<td>O30, TP21</td>
<td>location mode change</td>
<td>switch-off</td>
</tr>
<tr>
<td></td>
<td>O31, TP22</td>
<td>location mode change</td>
<td>switch-on</td>
</tr>
<tr>
<td></td>
<td>O12, TP19</td>
<td>location mode change</td>
<td>set thermostat heating</td>
</tr>
<tr>
<td></td>
<td></td>
<td>location mode change</td>
<td>set thermostat cooling</td>
</tr>
</tbody>
</table>

TP = Third-party, O = Official
S = General properties
P = App-specific properties
Soteria – A system for formal verification of IoT apps through model checking

**Source code**

```python
section("Turn on a pump...") {
    input "valve_device", "capability.valve", title: "Which?",
    required: true }

def installed() {
    subscribe(valve_device, "water.wet", waterWetHandler)
}
```

**IR**

// Permissions block
input (water_sensor, waterSensor, type:device)
input (valve_device, valve, type:device)

**State-model**

```
[water.dry, valve.close]  
[water.wet, valve.close]  
[water.dry, valve.open]   
[water.wet, valve.open]   
```

**Model Checking**

**Property**

```
water.wet ⇒ (AX valve.on)
```

**SMV format of the state-model**

**Output**

Stacktrace

Using NuSMV symbolic model checker...

General properties failed at state-model construction: none

NuSMV >> read model ...

NuSMV >> check property

NuSMV >> true
IoTBench

V.1.0.1 Released May 2018

IoTBench-test-suite
A micro-benchmark suite to assess the effectiveness of tools designed for IoT apps

- iot-platform
- smartthings
- openhab
- malicious-behaviors
- data-leaks

Groovy  ★ 10  2  Updated on May 12

https://github.com/IoTBench/

27 data leaks
28 security/safety violations
15 attacks migrated from mobile phone security
500+ official and third party apps
Thank you for listening!