

Conflict Verification in Connected IoT Apps

Aqsa Kashaf
Anand K. Prakash
Daniel Calderon

ConViCT

Motivation

App2: When there is smoke, play fire alarm

App1: Play music at 7 pm when I am home

User Specified IoT Apps can conflict leading to security implications

Conflicts

Direct Conflicts Example: App 3 & 4

Indirect Conflicts Example: App 1 & 2

App 3: When temperature > 70, open windows.

App 4: When nobody is in the room, close windows

Overview

SmartThings

Third Party Apps

SmartThings App Source Code

Smart Devices

Hub

Conflict?

ConViCT: Static Analysis Tool to Verify Apps Against Invariants

Challenges

- Complexity of IoT Apps
 - 1) Stateless
 - 2) Discrete Stateful
 - 3) Continuous Stateful (Time)
- Defining a unified model that covers the above mentioned categories of app, achieved by using timed automata.
- Parsing and Compilation of groovy code into timed automata.

System

Samsung SmartThings Apps (Groovy)

Parser

Model: Timed Automata of Input Apps

Preprocessing

Uppaal

Conflict Invariants

Library for Indirect Conflicts

Conflict yes/no

Illustration

Legend: ● Clock Invariant ● Update ● Guard ● Input Channel

App 1: Turn A/C on for 1 hour if main door is opened

App 2: If meter reading > threshold, turn off A/C

$A \models \rho$ → **property not satisfied**

$\rho = \text{not}(\text{output1.device} == \text{output2.device} \text{ and } \text{output1.value} != \text{output2.value})$

The above figures show the parser output of two Samsung SmartThings apps with their description. This output is given to Uppaal along with the conflict invariant described as ρ . Uppaal then composes the two automata in parallel and verifies the property.

Future Work

- Provide an API to users to define their own security invariants to verify against our tool.
- Develop a framework for conflict resolution.
- Evaluate the performance of the tool in terms of soundness and completeness.