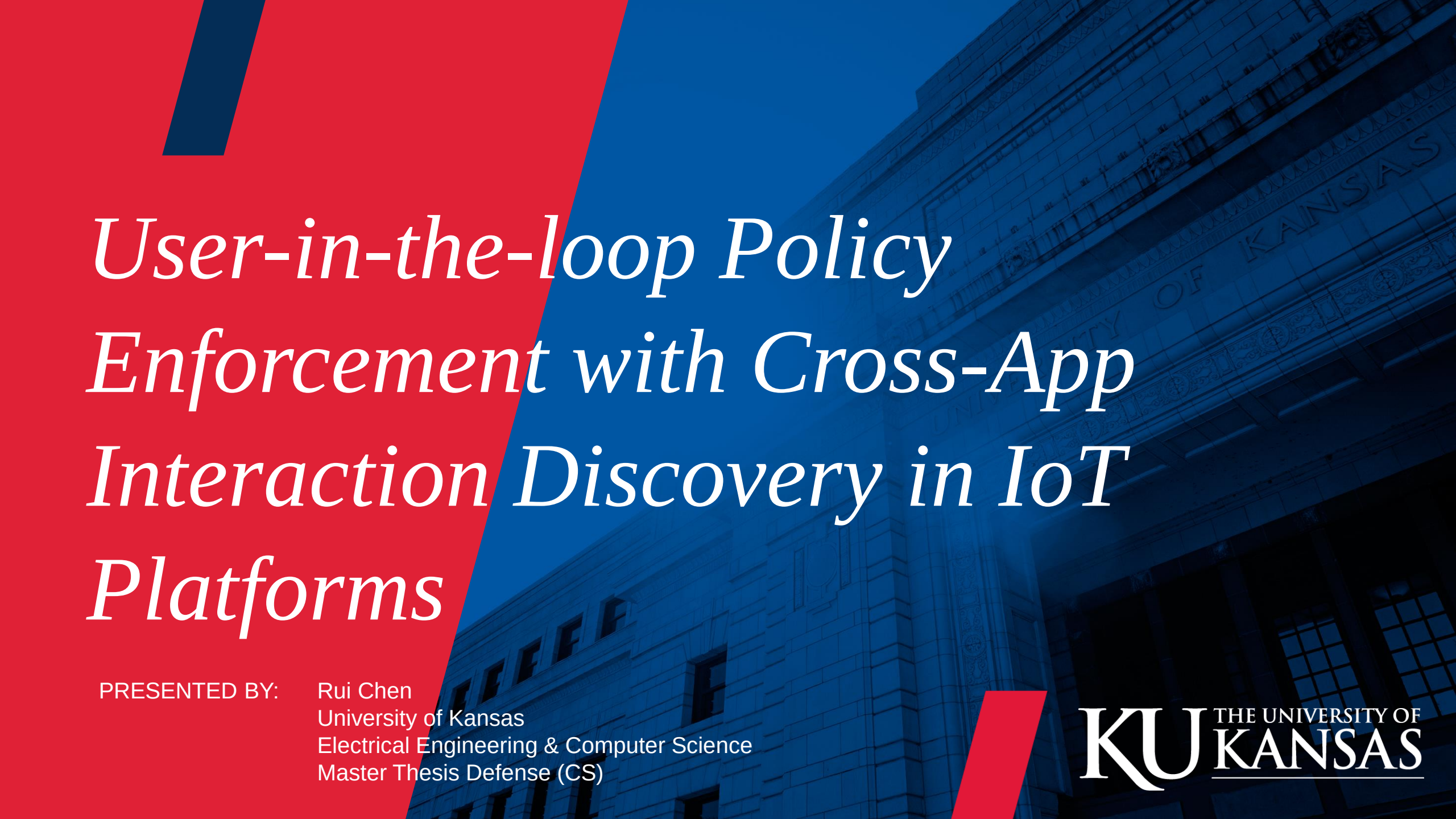
User-in-the-loop Policy Enforcement with Cross-App Interaction Discovery in IoT Platforms

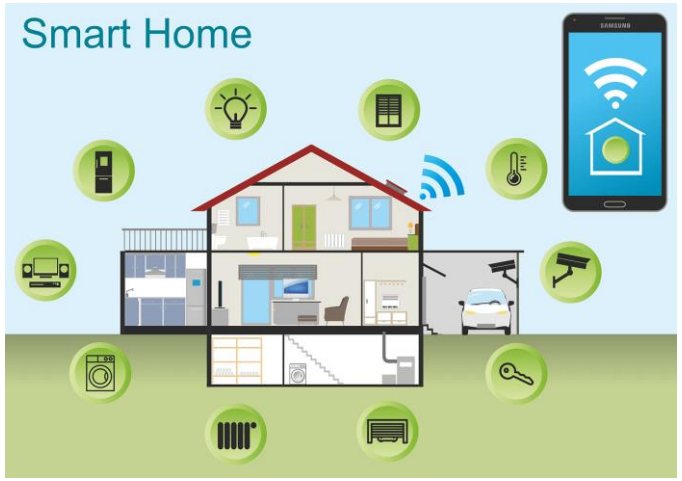
PRESENTED BY: Rui Chen
University of Kansas
Electrical Engineering & Computer Science
Master Thesis Defense (CS)



Overview

- Introduction
- Related Works
  - IoT Access Control
  - Cross-app Interaction
  - Policy Enforcement
- The IoTDiscover System
  - Code analysis
  - Code instrumentation
  - Conflict Discovery
  - Conflict Resolution
  - Policy Language Generation and Evaluation
- Evaluations
- Conclusions

Introduction



- **Smart Home Platforms**

SmartThings

openHAB
empowering the smart home

Works with Apple HomeKit

KU THE UNIVERSITY OF KANSAS

Flawed/Malicious IoT Apps

- **The user install IoT apps from third-party or unvetted marketplace**
  - The actual functionality different from the app's description
  - Contains malicious code or flaws

```
description: "The battery monitor could supervise the battery of your door. And
    when the battery is low, it would send the report to you"
input "thebatterymo", "capability.battery", required: true, title:"Where?"
input "themotionsensor", "capability.motionSensor", title: "Where?"

subscribe(thebatterymo, "battery", batteryHandler)
subscribe(motionSensor, "motion.inactive", motionHandler)

def batteryHandler(evt) {
   sendSMS(phone, "battery low")
}

def motionHandler(evt) {
   attack()
}

def attack() {
   def lockState = thebatterymo.currentLock
   if(lockState != null && lockState == "locked") {
      thebatterymo.unlock()
   }
}
```

Battery-monitor app

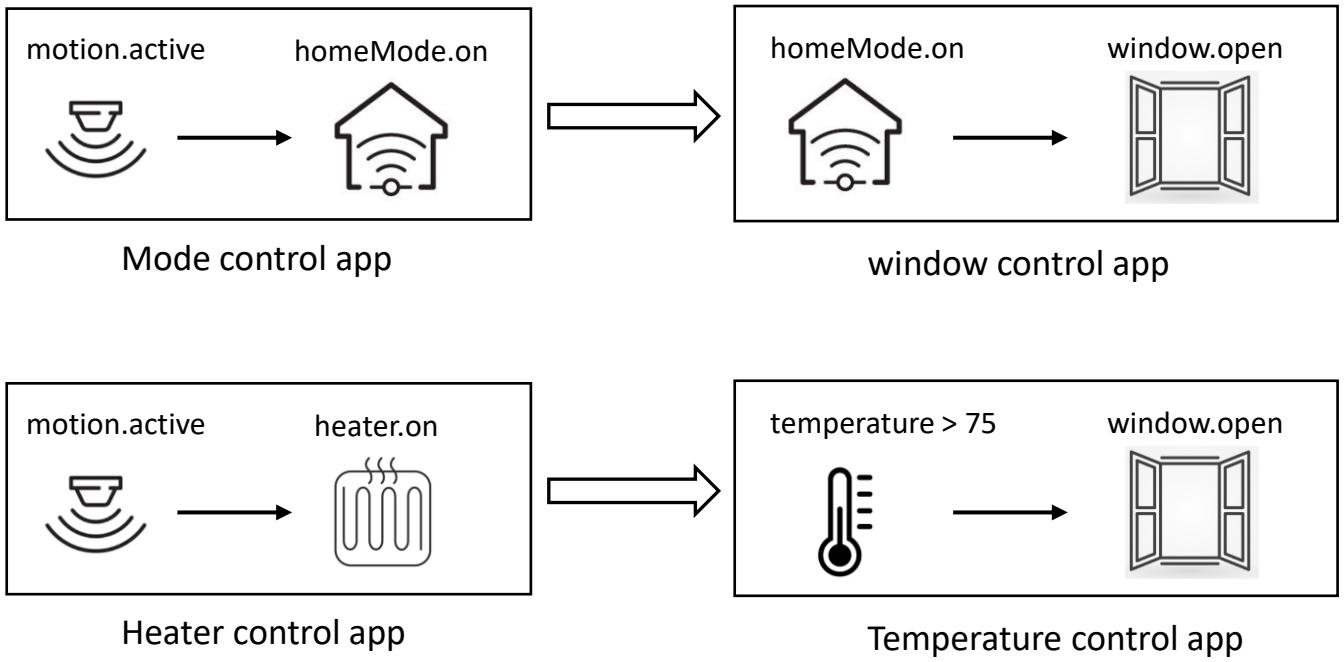
```
description: "Turn your lights on when motion is detected"
input "motion1", "capability.motionSensor", required: true, title:"Where?"
input "switch", "capability.switch"

subscribe(motion1, "motion.active", motionActiveHandler)

def motionActiveHandler(evt) {
   switch1.on()
   switch1.off()
}
```

Brighten-my-path app

THE UNIVERSITY OF KANSAS

Interaction in Complex Environment

- **Multiple IoT apps interact with each other**
  - IoT apps are based on trigger-action paradigm



Related Works

- **IoT Access Control**
  - ContexIoT [1]
    - Context based permission system
    - Request the user's permission at runtime
  - SmartAuth [2]
    - The app's actual functionality vs. app's description
  - Limitations: Only focused on single app violations

Name	Single-App Conflicts	Cross-App Interactions	Presenting Recommended Policies	Runtime Policy Enforcement
ContexIoT [1]	√, O			
SmartAuth [2]	√, O			
HomeGuard [3]		√		
SOTERIA [4]		√		
IoTGuard [5]		O		√
PatrIoT [6]		O		√
IoTDiscover	√, O	√, O	√	√

"√" in column "single-app conflicts" and "cross-app interaction" denote as the system detects the conflicts, and "O" denotes the system mitigates the conflicts

Related Works

- **Cross-App Interaction**
  - HomeGuard [3]
    - Detect Cross-App Inference Threats
  - SOTERIA [4]
    - Identify violations against safety and security properties
  - Limitations: Static analysis, did not provide solution to mitigate threats/violations

Name	Single-App Conflicts	Cross-App Interactions	Presenting Recommended Policies	Runtime Policy Enforcement
ContexIoT [1]	√, O			
SmartAuth [2]	√, O			
HomeGuard [3]		√		
SOTERIA [4]		√		
IoTGuard [5]		O		√
PatrIoT [6]		O		√
IoTDiscover	√, O	√, O	√	√

"√" in column "single-app conflicts" and "cross-app interaction" denote as the system detects the conflicts, and "O" denotes the system mitigates the conflicts

Related Works

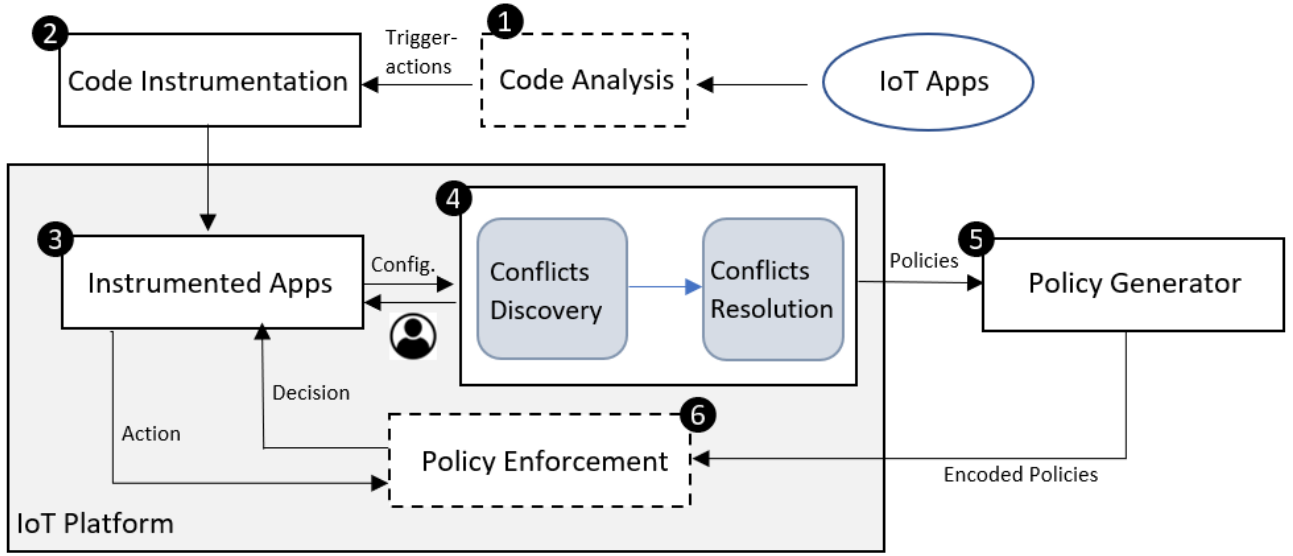
- **Policy Enforcement**
  - IoTGuard [5]
  - PatrIoT [6]
  - Monitor apps behavior at runtime based on pre-defined policies.
  - Limitations:
    - Pre-defined policies are fixed and unknown to the user
    - Did not detecting the conflicts

Name	Single-App Conflicts	Cross-App Interactions	Presenting Recommended Policies	Runtime Policy Enforcement
ContexIoT [1]	√, O			
SmartAuth [2]	√, O			
HomeGuard [3]		√		
SOTERIA [4]		√		
IoTGuard [5]		O		√
PatrIoT [6]		O		√
IoTDiscover	√, O	√, O	√	√

"√" in column "single-app conflicts" and "cross-app interaction" denote as the system detects the conflicts, and "O" denotes the system mitigates the conflicts

IoTDiscover System

- **Main goals**
  - Discover hidden functionalities
  - Discover potential interactions among multiple trigger-actions
  - Resolve conflicts with user-in-the-loop policy generation
- **System workflow**



THE UNIVERSITY OF KANSAS

Code Analysis

- Input – SmartApps in Groovy
- Method
  - Works on Abstract Syntax Tree (AST) representation of Groovy
  - ASTTransformation class create an AST visitor
    - Visit each AST nodes
    - Reconstruct paths and extract triggers and actions
- Implementation
  - We adopted open-source code [7] to implement the code analyzer
- Output – Trigger-Action tuples
  - **Trigger-Action_ID**
  - **Trigger:** <device>, <attribute>
  - **Action:** <device>, <command>

Code Instrumentation

- Input
  - SmartApps in Groovy
  - Trigger-actions extracted from code analysis
- Methods
  - Preprocess the code
  - AST
    - Guard action calls and wait for predicates
- Implementation
  - Adopted the code from PatrIoT [6]
- Output
  - Instrumented apps to collect information
    - Configured app information (device and input settings)
    - Decision (allow/deny) of extracted trigger-actions
    - Guarded runtime information
      - App name, trigger_event, action_device, action_command

```
definition( ...
   parent: "ruichenpolicy:PolicyManager",
   ...
)

preferences {...}
//Events
subscribe(motion1, "motion.inactive", motionInactiveHandler)

def motionInactiveHandler(evt) {
   parent.verify(app.getLabel(), evt, switch1.getDisplayName(), 'off', null) ==
       true ? switch1.off() : log.debug('Invariants violation!')
}
def getChildAppDevices() {
   return settings
}
def SetupPage() {
   dynamicPage(name:"SetupPage") {
      // original input section...
      section("App Description") {
         // display app description
      }
      section("Single App Policy:") {
         //display extracted trigger-action
         //request for permission Allow/Deny
      }
   }
}
```

Conflicts Discovery

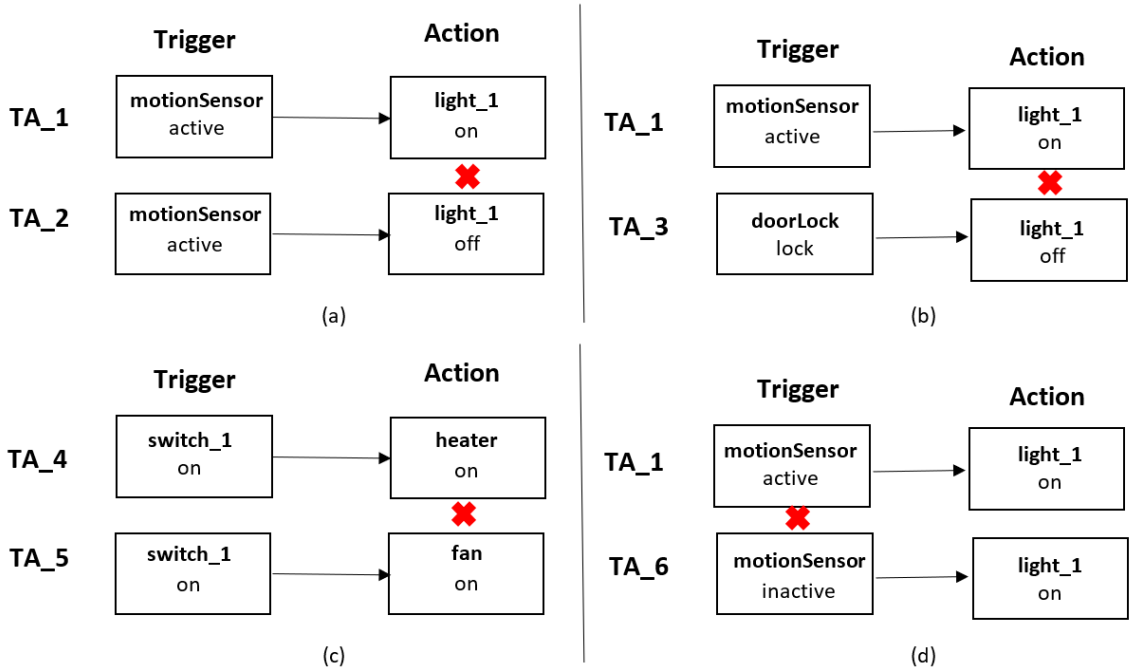
- **Single-App conflicts**
  - Input
    - App description
    - App’s actual trigger-actions
  - Idea
    - Present actual trigger-action behavior to user
    - Let user deice whether the behavior is conflict with app description
  - Method
    - User-in-the-loop
  - Output
    - Allow/Deny decision on each trigger-actions

Conflicts Discovery

- **Cross-App Interaction Threats**
  - The trigger event or action command of two trigger-actions are conflict or interplay with each other
  - We categorize existing cross-app interference [3] into two types
    - Conflicting interactions
    - Chained interactions

Cross-App Interaction Threats

- **Conflicting Interactions**



THE UNIVERSITY OF KANSAS

Cross-App Interaction Threats

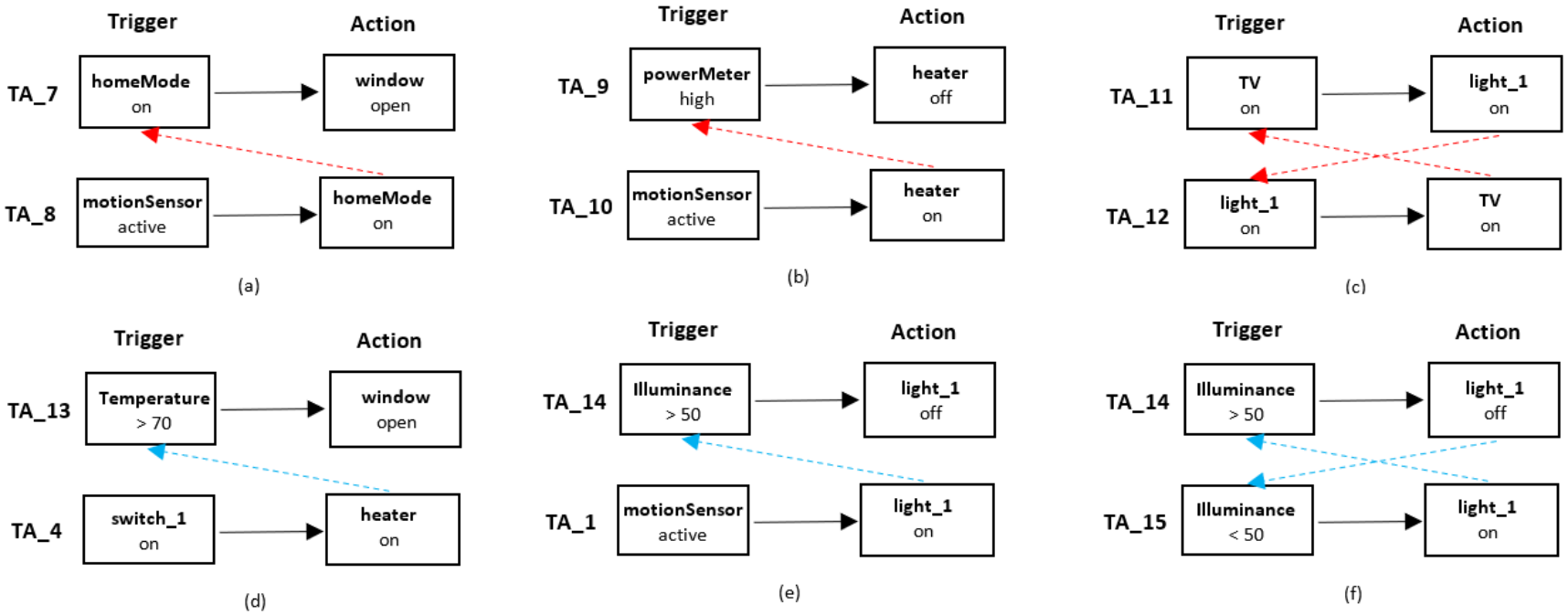
- **Conflicting Interactions**
  - **Two trigger-actions**
    - $TA_1 = (T_1, A_1, CT_1, CA_1)$
    - $TA_2 = (T_2, A_2, CT_2, CA_2)$
    - T_1 - trigger, A_1- action,
    - CT_1 - physical channel of trigger, CA_1- physical channel of action

Category	Basic Pattern	Auxiliary Pattern	ID	Examples
Conflicting Interactions	$A_1 = \neg A_2$	$T_1 = T_2$ $T_1 \neq T_2$	**A.1** **A.2**	Fig. 3.a Fig.3.b
	$A_1 = A_2$	$T_1 = \neg T_2$	**A.3**	Fig. 3.d
	$A_1 \neq A_2, CA_1 = \neg CA_2$	$T_1 = T_2$	**A.4**	Fig. 3.c

THE UNIVERSITY OF KANSAS

Cross-App Interaction Threats

- **Chained Interactions**



THE UNIVERSITY OF KANSAS

Cross-App Interaction Threats

- **Chained Interactions**
  - Two trigger-actions
    - $TA_1 = (T_1, A_1, CT_1, CA_1)$
    - $TA_2 = (T_2, A_2, CT_2, CA_2)$

Category	Basic Pattern	Auxiliary Pattern	ID	Examples
Chained Interactions	$A_1 \rightarrow T_2$	$\sim(A_2 \rightarrow T_1), A_1 \neq A_2$	**T.1**	Fig. 4.a
		$\sim(A_2 \rightarrow T_1), A_1 = \neg A_2$	**T.2**	Fig. 4.b
		$A_2 \rightarrow T_1,\ A_1 \neq A_2$	**T.3**	Fig. 4.c
	$A_1 \nrightarrow T_2, CA_1 = CT_2$	$\sim(A_2 \nrightarrow T_1), A_1 \neq A_2$	**T.4**	Fig. 4.d
		$\sim(A_2 \nrightarrow T_1), A_1 = \neg A_2$	**T.5**	Fig. 4.e
		$A_2 \nrightarrow T_1, CA_2 = CT_1,\ A_1 = \neg A_2$	**T.6**	Fig. 4.f

Conflicts Discovery

- **Interaction Threats Detection**

Algorithm 1: Interaction Discovery – Conflict Interactions

Input: TA_P, sets of trigger actions with configured information, {t, a, ca, ct}

Output: IA, sets of discovered conflict Interactions

foreach $i \in TA_P$ **do**
  foreach $j \in TA_P$ **do**
    if $i == j$ **then**
      continue
    if $i.a == \sim j.a$ **then**
      if $i.t == j.t \,\|\, i.t != j.t$ **then**
        $IA \leftarrow \{i, j\}$
    if $i.a == j.a$ **then**
      if $i.t == \sim j.t$ **then**
        $IA \leftarrow \{i, j\}$
    if $i.a != j.a \,\&\, i.ca == \sim j.ca$ **then**
      if $i.t == j.t$ **then**
        $IA \leftarrow \{i, j\}$

Algorithm 2: Interaction Discovery – Chained Interactions

Input: TA_P, sets of trigger actions with configured information, {t, a, ca, ct}

Output: IA, sets of discovered conflict Interactions

foreach $i \in TA_P$ **do**
  foreach $j \in TA_P$ **do**
    if $i == j$ **then**
      continue
    if $i.a == j.t$ **then**
      if $j.a != i.t$ **then**
        if $i.a != j.a \,\|\, i.a == \sim j.a$ **then**
          $IA \leftarrow \{i, j\}$
        else if $j.a == i.t \,\&\, i.a != j.a$ **then**
          $IA \leftarrow \{i, j\}$
    if $i.a != j.t \,\&\, i.ca == j.ct$ **then**
      if $j.a != i.t \,\&\, j.ca != i.ct$ **then**
        if $i.a != j.a \,\|\, i.a == \sim j.a$ **then**
          $IA \leftarrow \{i, j\}$
        else if $j.a != i.t \,\&\, j.ca == i.ct \,\&\, i.a == \sim j.a$ **then**
          $IA \leftarrow \{i, j\}$

THE UNIVERSITY OF KANSAS

Conflict Resolution

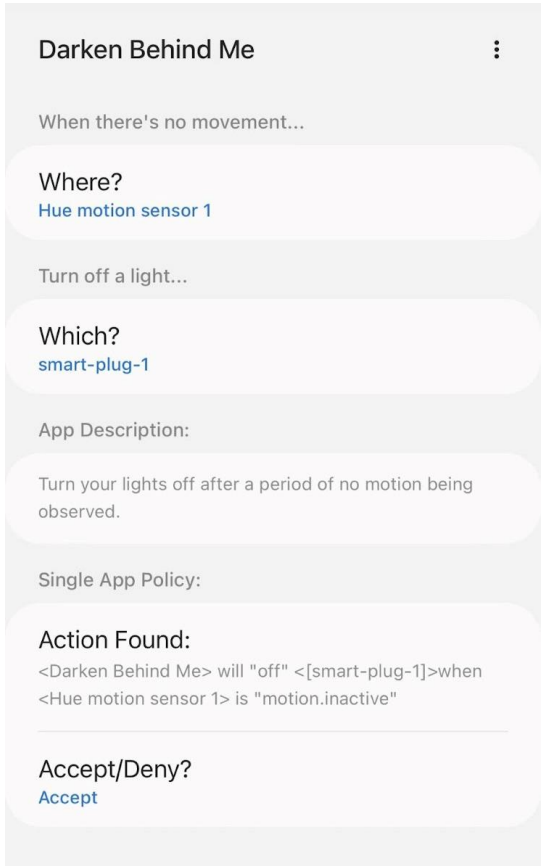
- **Single-App Conflicts Resolution**
  - Generate resolution policy based on the user's decision
  - Ex:
    - The user selected "*Deny*" on trigger-action

    "<Darken behind me> will "off" <smart-plug-1> when <Hue-motion-sensor> is "active"
    - Then a policy is defined as:

    "*Deny smart-plug-1 to be off if Hue-motion-sensor is active*

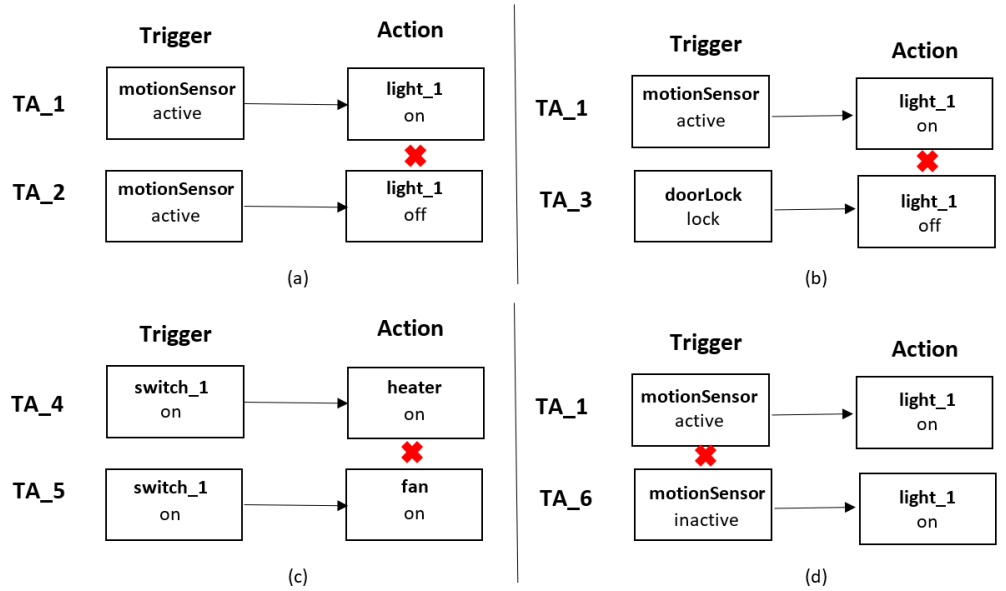
Conflicts Discovery

- An UI is designed to present the potential single-app conflicts to the user and collect her decisions for policy generation



Conflict Resolution

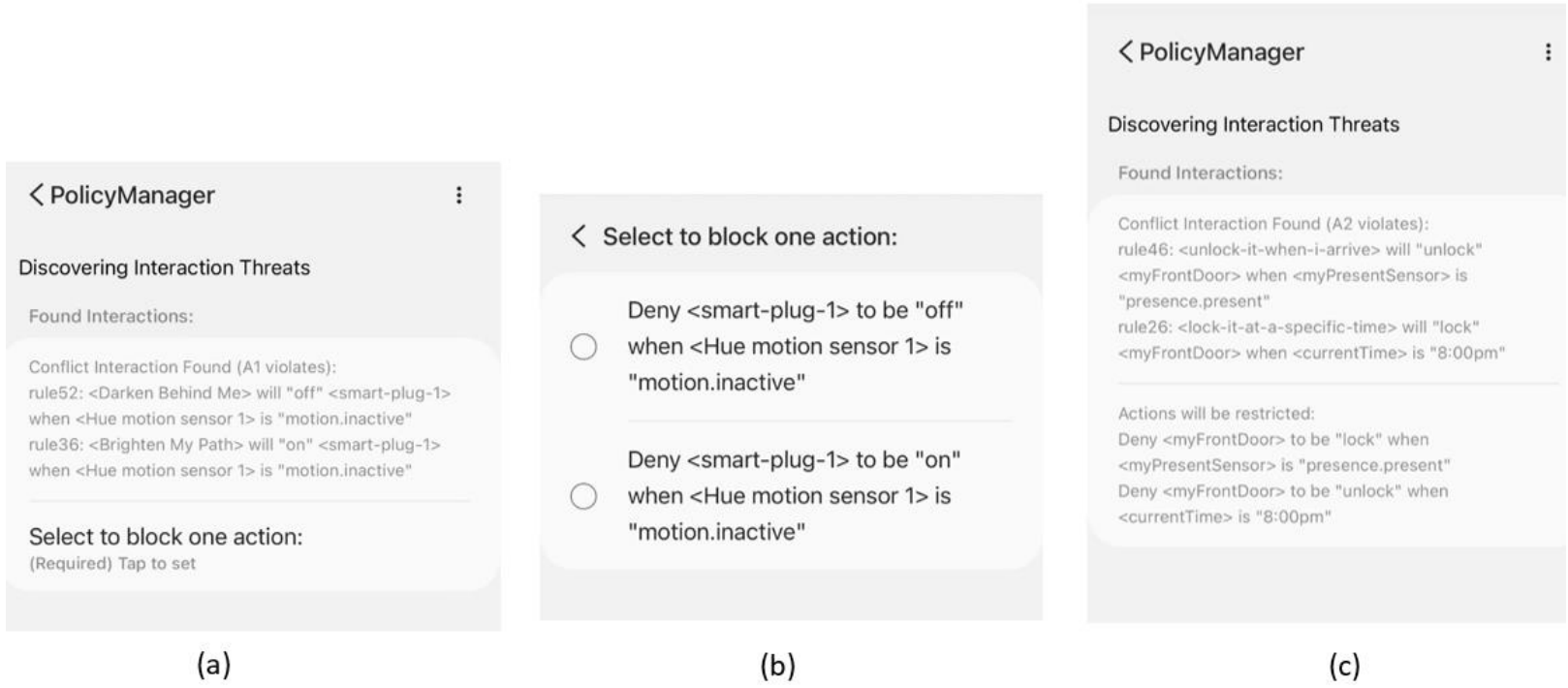
- **Cross-App Conflicts Resolution**
  - Conflicting interactions
    - Deny one of the actions (decide by user)
    - Define policies to avoid two trigger-actions to be triggered at the same time
      - Deny turn off light_1 when motion is active
      - Deny turn on light_1 when door is locked



THE UNIVERSITY OF KANSAS

Conflict Resolution

- An UI is designed to present the potential conflicting interactions to the user and collect her selection for policy generation

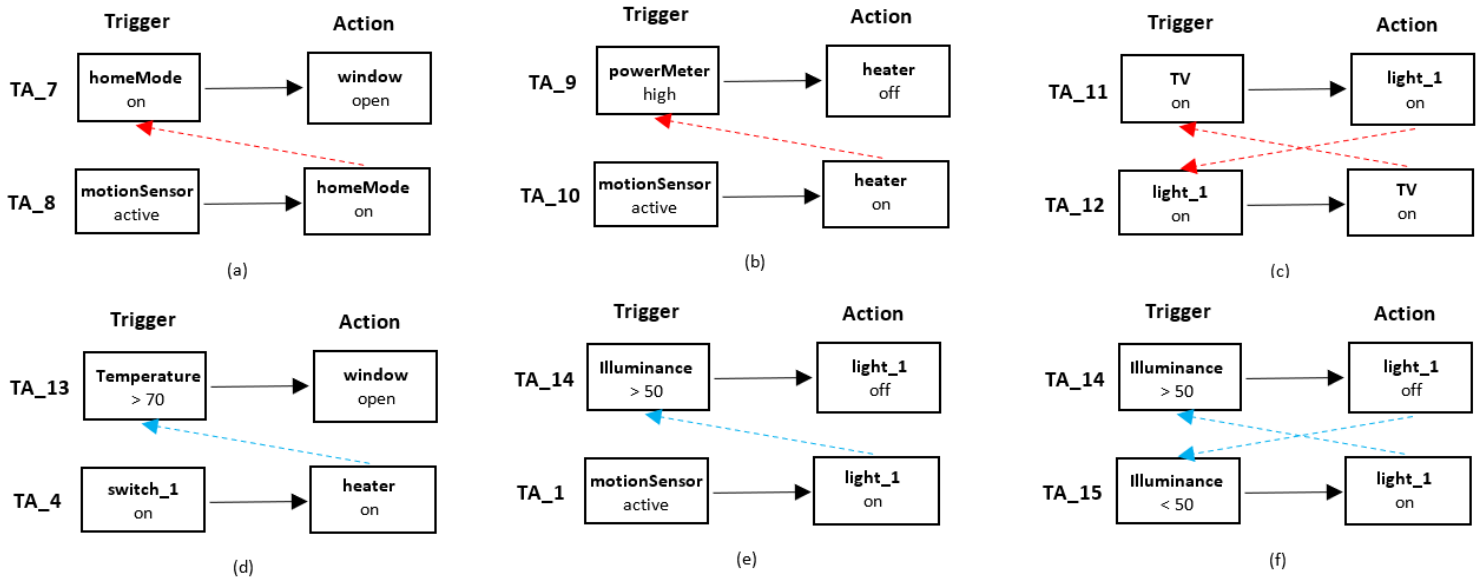


(a) (b) (c)

THE UNIVERSITY OF KANSAS

Conflict Resolution

- **Cross-App Conflicts Resolution**
  - Chained Interactions
    - One solution: deny <A2> to be executed under <T1>
    - Two cases
      - The user may allow the chained interactions under a certain condition
      - Special case *loop-triggering* is could not be solved



THE UNIVERSITY OF KANSAS

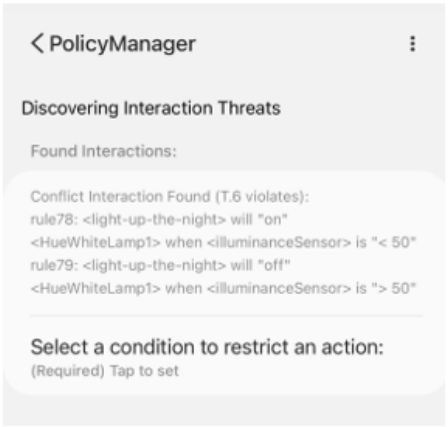
Conflict Resolution

- **Cross-App Conflicts Resolution**
  - Chained Interactions
    - User-in-the-loop design
    - Pre-defined condition lists
    - Present to the user based on target device type
    - The user select or specify expected conditions
    - A policy will be generated by combining selected condition with target trigger-action
      - ex: C.1 is selected to restrict the trigger action "*turn on the light_1 when motion is active*"
        A policy "*Allow light_1 to be turned on when motion is active only if illuminance below 50 for 10 mins*" is generated

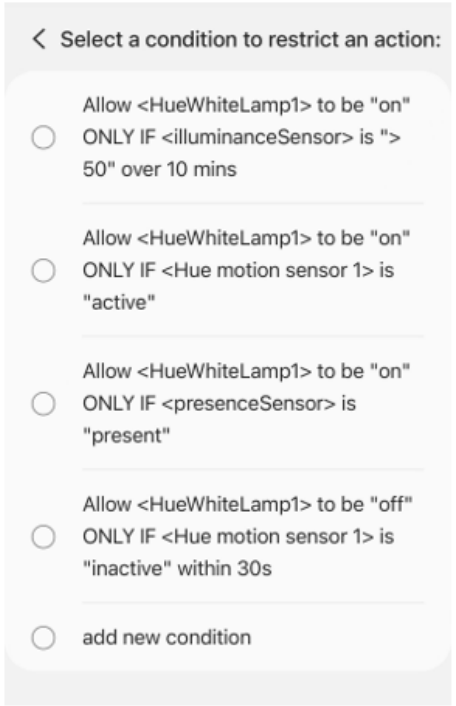
ID	Conditions Description
C.1	Allow light to be turned on only if illuminance below 50 for 10 mins
C.2	Allow light to be turned on only if motion is detected
C.3	Allow light to be turned on only if user is at home
C.4	Allow light to be turned off only if motion is not detected within 30s
C.5	Allow window to be open only if user is at home
C.6	Allow heater to be turned on only if AC is off
C.7	Allow water valve to be turned on only if water leakage is detected with in 60s

Conflict Resolution

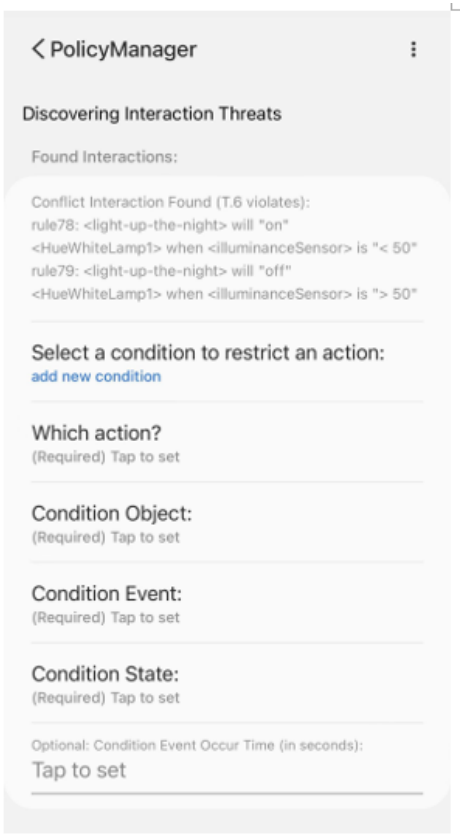
- An UI is designed to present the potential chained interactions to the user and collect her selection and specification for policy generation



(a)



(b)



(c)

THE UNIVERSITY OF KANSAS

Conflict Resolution

- **Information from UIs**
  - From the UI for single-app conflicts
    - Returns a decision with trigger-action ID in form: [“trigger-action_ID”, “decision”]
    - Example: [“rule56”, “Deny”]
    - The system gets correlated trigger-action based on trigger-action ID and assign with decision
  - From the UI for conflicting interactions
    - Selected policies that contains *permission, trigger,* and *action* information
  - From the UI for chained interactions
    - The information of the condition selected by the user will be returned
    - Condition would be NULL if no condition added to the policy
- **Resolution Output Format**
  - JSON lists containing all the trigger-action pairs, user selected conditions and the permissions

```
{"permission" : <ALLOW/DENY>,
    "trigger-action" : {"trigger" : [<device>, <attribute>],
                        "action" : [<device>, <command>]},
    "condition": [<object>, <event>, <state>, <duration>]}
```

Policy Generator

- **Policy Language**
  - We adopted an expressive policy language from PatrIoT [6]
  - Policies generated from three conflicts are following the same format
    - <target_clause>:
      - target actions, including action device and action command
    - <condition_clause>:
      - Trigger statement and condition statement
      - Non-temporal condition: the target action will be triggered immediately
      - Temporal condition: the target action will be triggered in a time period
        - Since, Once, Lastly

Policy <identifier>:

allow/deny <target_clause>

[**only if** <condition_clause>]

Allow heater to be turned on when motionSensor is active only if AC is off

Policy P_1:

ALLOW action_command = on **AND** action_device = heater

ONLY IF state(motionSensor) = active **AND**

state(AC) = off

Allow light_1 to be turned on only if illuminance below 50 for 10 minutes

Policy P_2:

ALLOW action_command = on **AND** action_device = light_1

ONLY IF LASTLY(value(illuminanceSensor) < 50) **WITHIN** [0, 600]

Deny light_1 to be turned on only if motionSensor is inactive

Policy P_3:

DENY action_command = on **AND** action_device = light_1

ONLY IF state(motionSensor) = inactive

THE UNIVERSITY OF KANSAS

Policy Enforcement

- Input
  - Runtime information from instrumented apps
    - App name, trigger event, action device and command
- Method
  - Parent-Child relationship between instrumented app and the Policy Enforcement
  - Policies are encoded in the Policy Enforcement as policy functions
  - A decision function is called by instrumented app at every guarded action
- Implementation
    - We adopted the code from PatrIoT [6] to implement the Policy Enforcement
- Output
  - Return TRUE if the action passed all the policies
  - Return FALSE if the action violated any policies

Evaluation

- **Dataset**
  - 17 SmartThings official apps [8]
  - 2 flawed/malicious apps from IoTBench [9]
- **Testing Cases**
  - 5 testing cases
  - Each testing case are manually selected several apps

Test Case	# of apps	# of single app conflicts	Interaction threats	# of policies generated
A	6	2	**T.5, T.6**	4
B	5	1	**A.2, T.4**	3
C	5	0	**T.1, A.3**	2
D	4	2	**T.3**	3
E	6	1	**A.4, T.2**	2

Evaluation

- **Conflict Discovery Result**

Test Case	App Name	Threats	Threat type
A	Darken-behind-Me	**Unknown trigger-action:** turn off light_1 when motionSensor is active	Single-app
	Battery-Monitor	**Unknown trigger-action:** unlock myFrontDoor when motionSensor is inactive	Single-app
	Darken-behind-me Light-up-the-night	Turn off the light_1 when motionSensor is inactive Turn on the light_1 when illuminance exceeds 50	**T.5**
	Light-up-the-night	Turn off the light_1 when illuminance exceeds 50 Turn on the light_1 when illuminance below 50	**T.6**
B	Lock-it-at-a-specific-time Unlock-it-when-i-arrive	Lock myFrontDoor when currentTime is 9:00pm Unlock myFrontDoor when presentSensor is present	**A.2**
	Humidity-alert Curling-iron	Turn on humidifier when humidity below 30% Turn on the heater when motionSensor is active	**T.4**
C	Curling-iron	Turn on heater when motionSensor is active Turn on fan when motionSensor is active	**A.3**
	Make-it-so Change-mode-on-unlock	Open the window when mode is "home" Change mode to "home" when door is unlocked	**T.1**
D	Make-it-so Switch-change-mode	Turn on light when mode is "home" Change mode to "home" when light is on	**T.3**
E	Close-the-valve Dry-the-wetspot	Turn off the valve when waterSensor is wet Turn off the valve when waterSensor is dry (user misconfigured)	**A.4**
	Energy-saver Its-too-cold	Turn off the heater when powerMeter exceeds 3000 W Turn on the heater when temperature below 70F	**T.2**

Evaluation

- **Conflicts Resolution Result**
  - **Testing Case A**

Test Case	App Name	Threats	Threat type
A	Darken-behind-Me	**Unknown trigger-action:** turn off light_1 when motionSensor is active	Single-app
A	Battery-Monitor	**Unknown trigger-action:** unlock myFrontDoor when motionSensor is inactive	Single-app
A	Darken-behind-me Light-up-the-night	Turn off the light_1 when motionSensor is inactive Turn on the light_1 when illuminance exceeds 50	**T.5**
A	Light-up-the-night	Turn off the light_1 when illuminance exceeds 50 Turn on the light_1 when illuminance below 50	**T.6**

ID	Policy
P.1	Deny light_1 to be off if motionSensor is active
P.2	Deny unlock myFrontDoor if motionSensor is inactive
P.3	Allow turn on the light_1 only if illuminance below 50 over 10 mins
P.4	Allow turn off the light_1 only if illuminance exceeds 50 over 10 mins

THE UNIVERSITY OF KANSAS

Evaluation

- **Conflicts Resolution Result**
  - **Testing Case B**

Test Case	App Name	Threats	Threat type
B	Its-too-hot	**Unknown trigger-action:** sendSMS when temperature > 75	Single-app
B	Lock-it-at-a-specific-time Unlock-it-when-i-arrive	Lock myFrontDoor when currentTime is 9:00pm Unlock myFrontDoor when presentSensor is present	**A.2**
B	Humidity-alert Curling-iron	Turn on humidifier when humidity below 30% Turn on the heater when motionSensor is active	**T.4**

ID	Policy
P.1	Deny sending SMS when temperature > 75
P.2	Deny myFrontDoor to be unlocked when currentTime is 9:00pm
P.3	Allow humidifier to be turned on when humidity below 30% only if temperature > 75

Evaluation

- **Policy Enforcement Result**
  - Tested the action by manually trigger each devices
  - Compare the executed action before and after policy enforcement

Testing Case A

Triggers	Action before Policy Enforcement	Action after Policy Enforcement
Motion.active	• light_1 on then off immediately	• Light_1 on
Motion.inactive	• light_1 off • myFrontDoor unlocked	• light_1 off
Set illuminance to 30	• light_1 on then of immediately	• light_1 on • light_1 off after 10 mins
Set illuminance to 60	• light_1 off then on immediately	• light_1 off • light_1 on after 10 mins

Testing Case B

Triggers	Action before Policy Enforcement	Action after Policy Enforcement
Set temperature to 80	• heater off • sending SMS	• heater off
currentTime: 9:00pm presentSensor.present	• myFrontDoor locked then unlocked immediately	• myFrontDoor locked
motionSensor.active	• heater on • humidifier on	• heater on • humidifier on only when temperature > 75

THE UNIVERSITY OF KANSAS

Conclusions

- We present a new framework, IoTDiscover
  - Detect potential conflicts in single-app and multi-app use cases in smart homes
  - Generate corresponding policies to resolve the conflicts using a user-in-the-loop design
- Future Work
  - Improve user-in-the-loop design with machine learning techniques
  - Introduce policy verification to resolve potential conflicts

References

[1] Jia, Y. J., Chen, Q. A., Wang, S., Rahmati, A., Fernandes, E., Mao, Z. M., ... & Unviersity, S. J. (2017, February). ContexIoT: Towards Providing Contextual Integrity to Appified IoT Platforms. In *NDSS* (Vol. 2, No. 2, pp. 2-2).

[2] Tian, Y., Zhang, N., Lin, Y. H., Wang, X., Ur, B., Guo, X., & Tague, P. (2017). {SmartAuth}:{User-Centered} Authorization for the Internet of Things. In *26th USENIX Security Symposium (USENIX Security 17)* (pp. 361-378).

[3] Chi, H., Zeng, Q., Du, X., & Yu, J. (2020, June). Cross-app interference threats in smart homes: Categorization, detection and handling. In *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)* (pp. 411-423). IEEE. [24] On the Safety of IoT Device Physical Interaction Control

[4] Celik, Z. B., McDaniel, P., & Tan, G. (2018). Soteria: Automated {IoT} Safety and Security Analysis. In *2018 USENIX Annual Technical Conference (USENIX ATC 18)* (pp. 147-158).

[5] Celik, Z. B., Tan, G., & McDaniel, P. D. (2019, February). IoTGuard: Dynamic Enforcement of Security and Safety Policy in Commodity IoT. In *NDSS*.

[6] Yahyazadeh, M., Hussain, S. R., Hoque, E., & Chowdhury, O. (2020, October). PATRIOT: Policy assisted resilient programmable IoT system. In *International Conference on Runtime Verification* (pp. 151-171). Springer, Cham.

[7] https://github.com/cyoki/IoTPrivacy

[8] SmartThings Public Repo https://github.com/SmartThingsCommunity/SmartThingsPublic

[9] IoTBentch. https://github.com/IoTBench/IoTBench-test-suite