CP-IoT: A Cross-Platform Monitoring System for Smart Home

作者：Hai Lin, Chenglong Li, Jiahai Yang, Zhiliang Wang, Linna Fan, Chenxin Duan

单位：Institute of Network Sciences and Cyberspace, BNRist, Tsinghua University

（清华网研院）

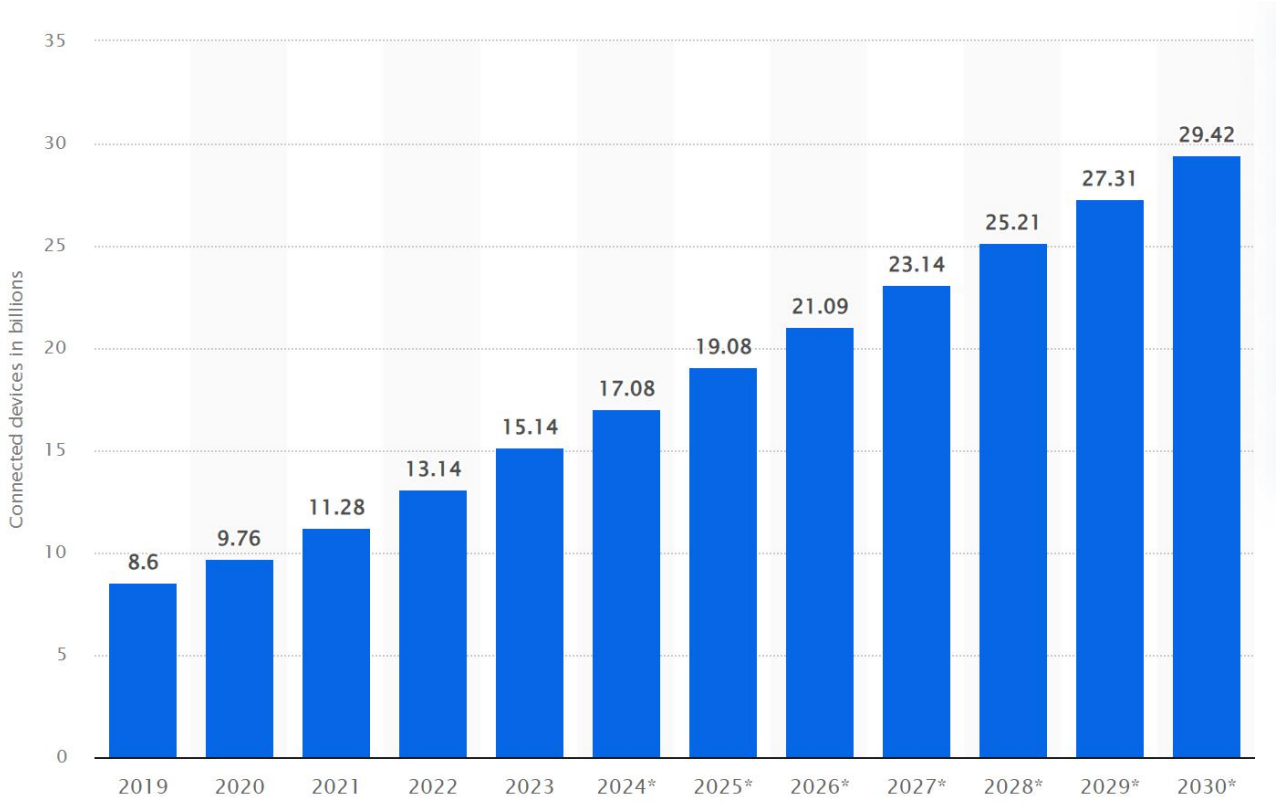
发表会议：NDSS 2024

允公允能 日新月异

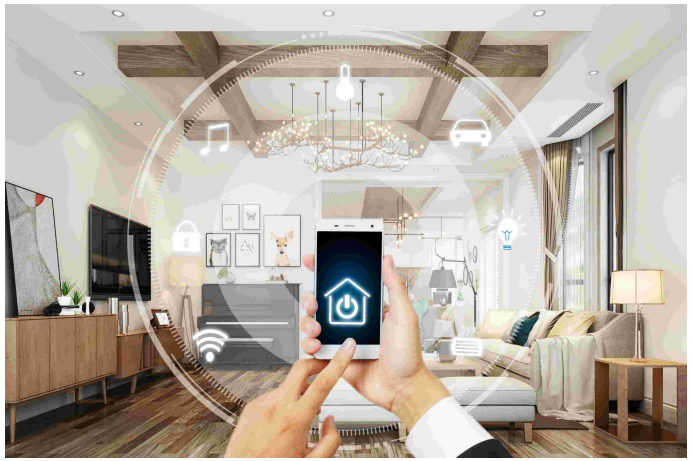
目录

01 研究背景与动机

研究背景与动机



智能物联网设备数量增长迅速



生活中大量使用智能设备

大量设备厂商

HomeKit

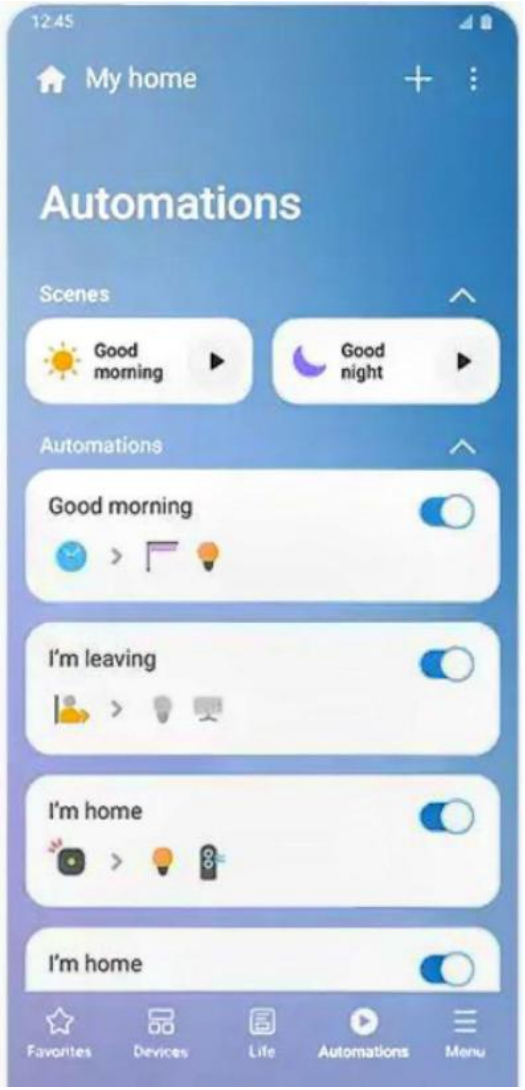
amazon alexa

SmartThings

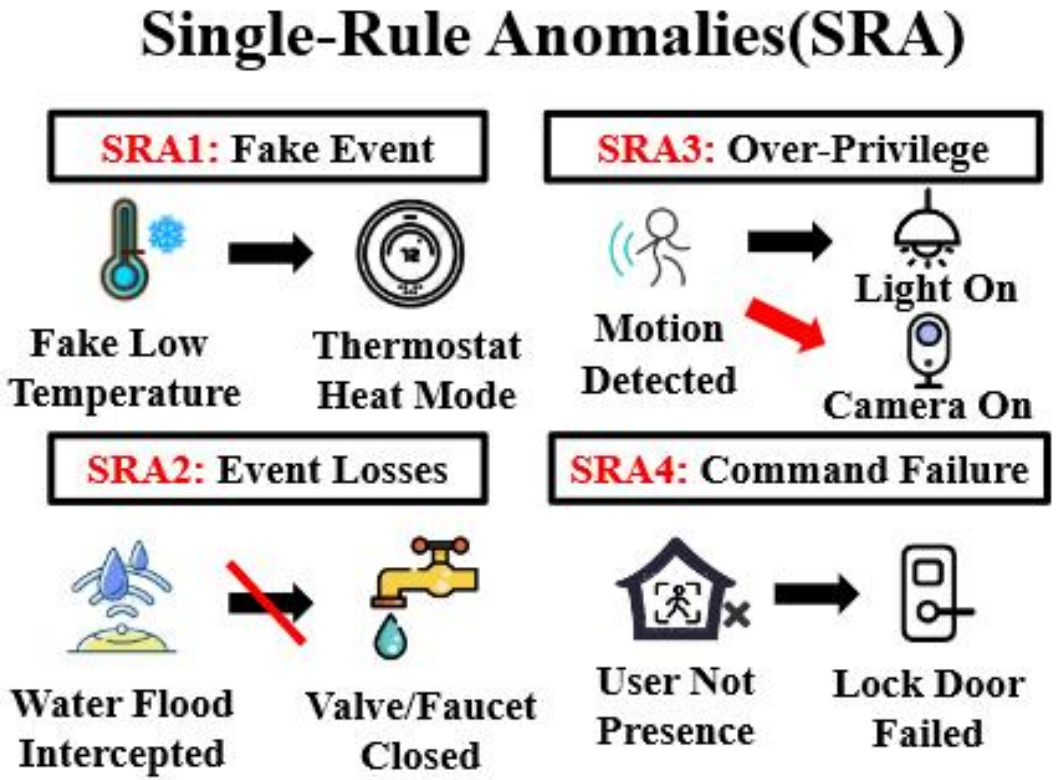
mijia

Huawei Smart Home

支持用户自定义场景/规则



规则存在问题

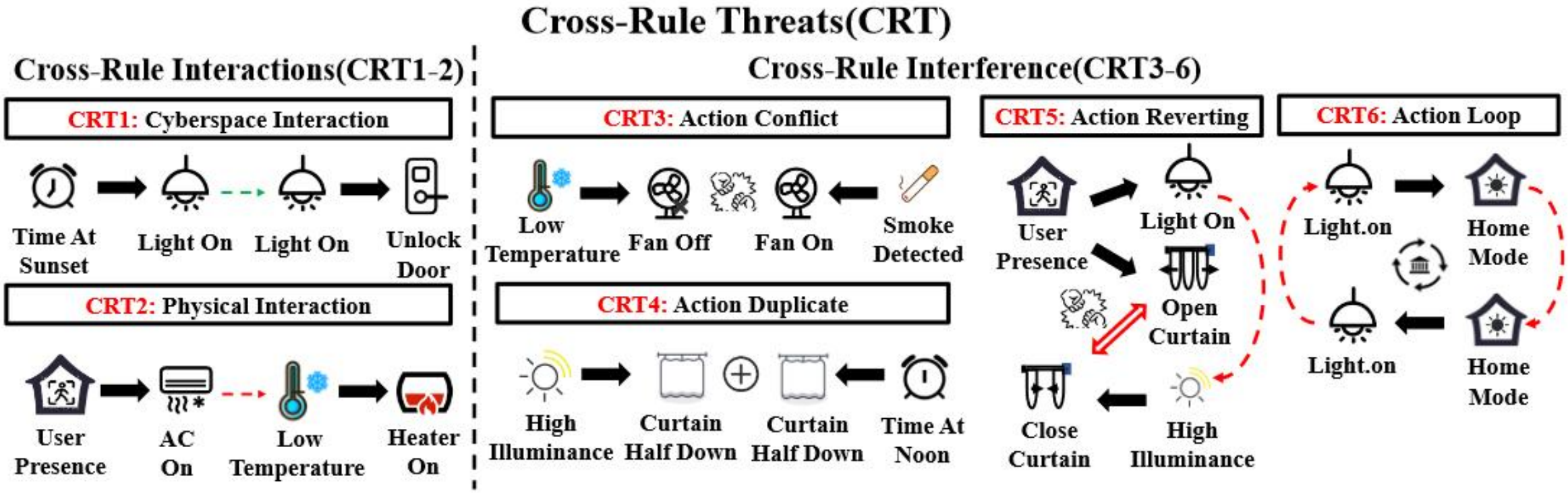


A study [1] finds 76 instances of 222 SmartThings apps are over-privilege, with 5.5% of related devices being misused.

A recent study [2] shows that more than 55% of SmartApps have extra permission to control unrelated devices.

[1] Atheer Abu Zaid, Manar H. Alalfi, and Ali Miri. Automated identification of over-privileged smartthings apps. 2019.
[2] Earlence Fernandes, Jaeyeon Jung, and Atul Prakash. Security analysis of emerging smart home applications. 2016.

规则存在问题


Cross-Rule Threats(CRT)
Cross-Rule Interactions(CRT1-2)
Cross-Rule Interference(CRT3-6)
CRT1: Cyberspace Interaction
Time At Sunset
Light On
Light On
Unlock Door
CRT2: Physical Interaction
User Presence
AC On
Low Temperature
Heater On
CRT3: Action Conflict
Low Temperature
Fan Off
Fan On
Smoke Detected
CRT4: Action Duplicate
High Illuminance
Curtain Half Down
Curtain Half Down
Time At Noon
CRT5: Action Reverting
User Presence
Light On
Open Curtain
Close Curtain
High Illuminance
CRT6: Action Loop
Light.on
Home Mode
Light.on
Home Mode

02 现有工作与挑战

已有分析方案

- Homonit[1]和IoTGaze[2] 通过对加密流量进行侧信道推断，获得自动化规则的执行行为进行建模，并通过上下文检查检测意外行为。
- HAWatcher[3]基于日志、设备配置等提取设备间的相关性。用影子设备(shadow device)检测运行时设备行为和相关性的偏差

- Homeguard[4]构建一个符号执行模块，从应用程序中提取自动化语义，并识别三种应用程序干扰威胁。
- IoTGuard[5]设计一个动态系统，收集应用程序的运行时信息，并将其存储在动态模型中。
- iRuler[6]将规则形式化，并应用SMT和模型检查

[1] Wei Zhang, Yan Meng, Yugeng Liu, Xiaokuan Zhang, Yinqian Zhang, and Haojin Zhu. Homonit: Monitoring smart home apps from encrypted traffic. 2018.

已有方案的问题

检测面窄

- 仅能检测单个自动化规则或跨规则威胁的不一致行为，但不能两者都检测
- 检测结果不充分，忽略了检测一些细粒度的子类型。

兼容性弱

- 大多数都是为SmartThings设计并开发基于代码的方法，用于规则提取/数据收集和威胁发现。
- 不同平台实现不同，现有方案无法检测其他平台的代码；
- 部分平台的代码框架不开源，比如苹果的Homekit
- 不同平台之间的日志格式和流量模式存在一些差异

已有方案能力范围

Method \ Support	SRA	CRT1-2	CRT3-6	Cross-platform
Homonit [14]	◐	○	○	○
HAWatcher [16]	◐	○	○	◐
IoTGaze [15]	●	●	○	◐
Soteria [17]	○	◐	◐	○
HomeGuard [20]	○	●	◐	○
IoTGuard [18]	○	◐	◐	◐
iRuler [19]	○	●	●	◐
CP-IoT	●	●	●	●

难点与挑战

1. 跨平台检测的挑战
    a) 如何从异构智能家居平台中提取自动化规则
    b) 如何识别不同平台规则的运行时行为
2. 防御各种威胁的挑战
    a) 如何保证检测结果的完整性
    b) 如何减轻这些威胁带来的风险

03 系统架构设计

挑战与见解

❶ 如何从异构智能家居平台中提取自动化规则

基于描述分析的配置页面的语义分析，这些分析在不同的平台之间共享。此外，它包含细粒度的规则定义和用户策略，两个阶段的结果可以相互补充，以达到较高的准确性。

❷ 如何识别不同平台规则的运行时行为

提取事件的包级指纹和流级指纹，包级指纹识别事件，流级指纹相区分平台。

将事件组合起来以获得每个规则的运行时行为

挑战与见解

❸ 如何保证检测结果的完整性

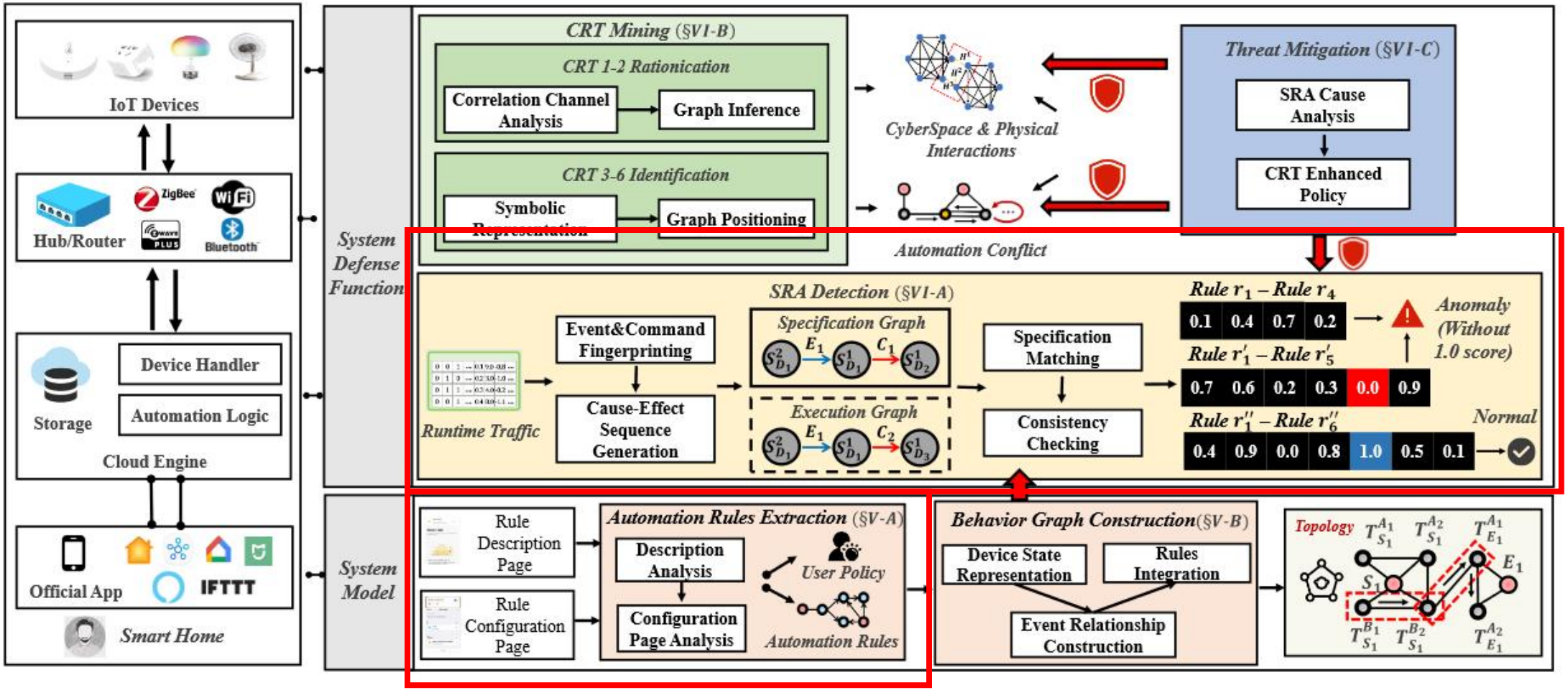
构建一个centralized的图作为行为模型。通过将每个设备建模为不同的节点，整合部署在不同平台上的所有自动化规则，可以在多个规则的控制下跟踪复杂的设备状态变化

❹ 如何减轻这些威胁带来的风险

“对症下药”

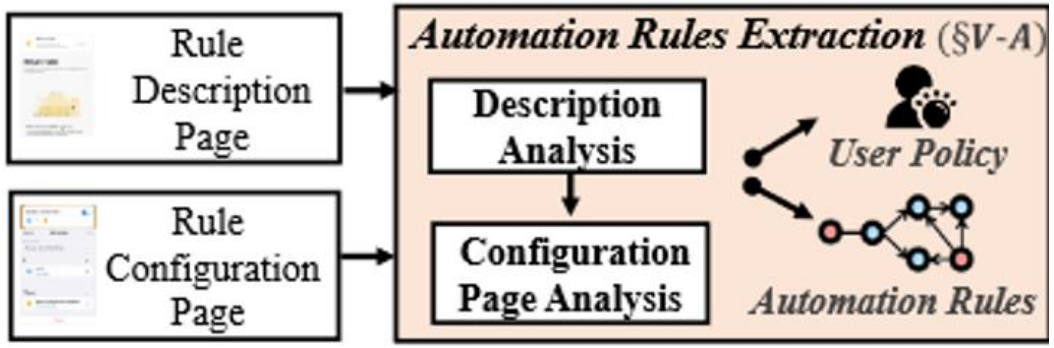
- 对于SRA：重复执行，区别是意外还是攻击
- 对于CRT：在两个以上的自动化之间添加安全策略，或者补充一些自动化规则来阻止危险的跨规则交互或干扰

总览



本文实现一个跨平台的监控系统*CP-IoT*，以确保自动化的正常工作，并发现不同自动化规则之间的高风险威胁。

自动化规则提取



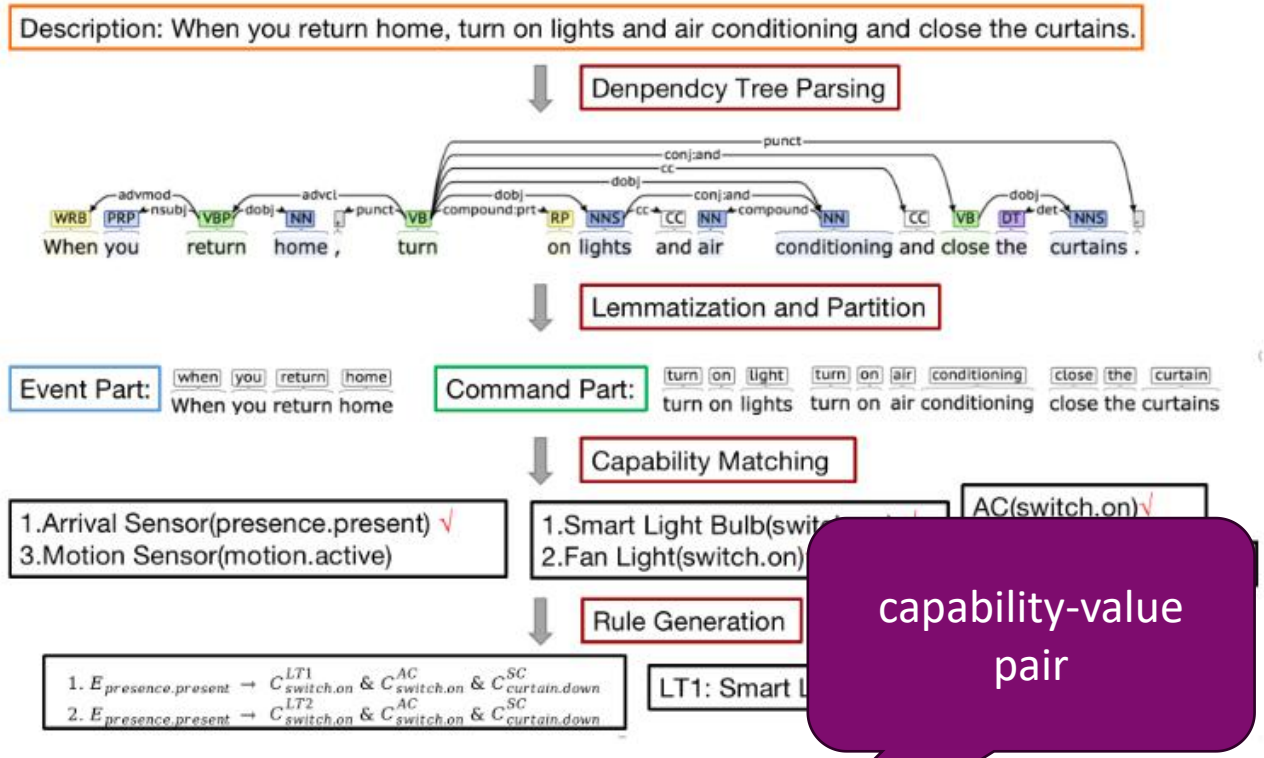
基于APP界面分析的规则提取

(对应Challenge 1)

工作流程

1. 规则描述分析
2. 配置界面分析

规则描述分析

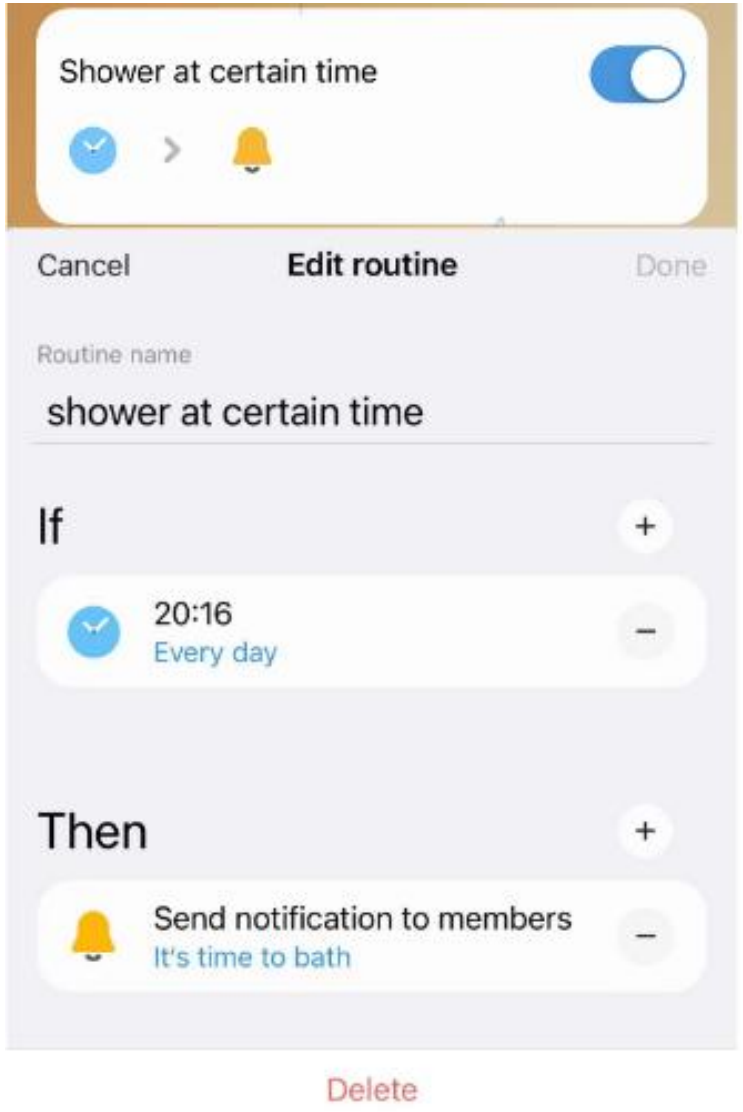


使用NLP工具StanfordCoreNLP[1]识别事件部分(E)和命令部分(C)。构建依赖树并解析，得到了每个单词的词性和单词之间的依赖关系。成对组合后获得每个事件和命令的最小表示。例如：turning on/off the light 可以抽象成“switch.on”/“switch.off”

抽象关键：相似度匹配。方案：使用BERT模型进行embedding，计算余弦相似度

[1] Stanfordcorenlp tool, 2022. https://github.com/stanfordnlp/CoreNLP

配置界面分析



- 将IF/Condition/Trigger/WHEN块的内容与规则的Event部分相匹配
- 将THEN/Action/Adjust块的内容与规则的Command部分相匹配。

符号表示

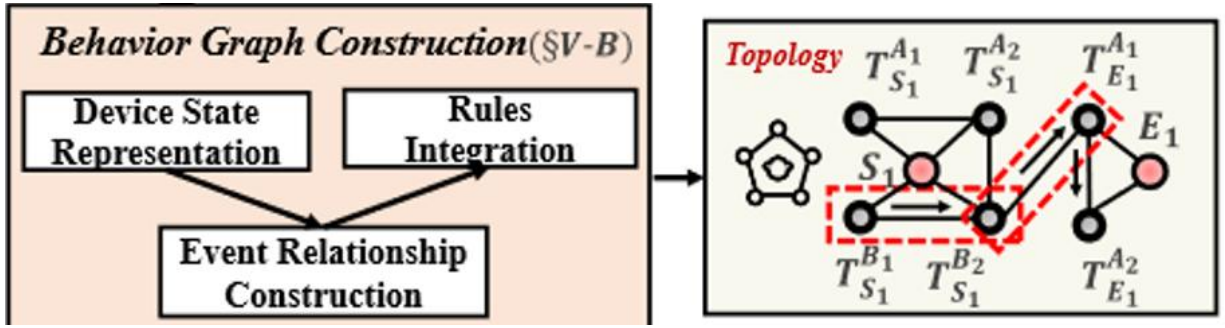
Scene Control - ξ II.A

- **Good Morning:** $U^{CA}_{curtain.up}$ & $U^{Mode}_{mode.home}$
- **Good Night:** $U^{CA}_{curtain.down}$ & $U^{Mode}_{mode.night}$

Automation Rule- ξ II.B

- **I'm Leaving:** $E^{AS}_{presence.not\ present} \rightarrow C^{LB}_{switch.off}$ & $C^{TV}_{switch.off}$
- **I'm Home:** $E^{AS}_{presence.present} \rightarrow C^{LB}_{switch.on}$ & $C^{LK}_{lock.unlock}$

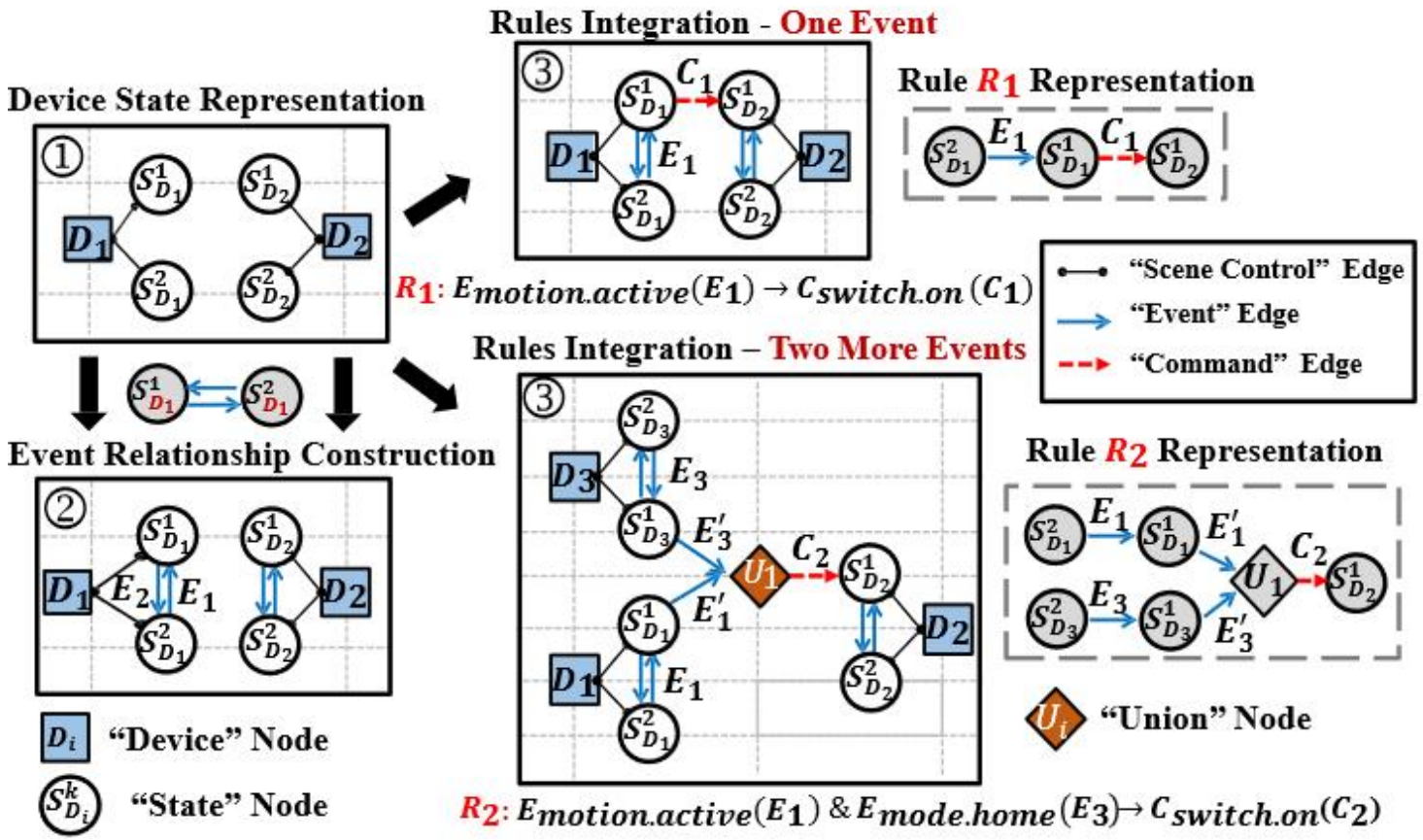
行为图构建



工作流程

1. 设备状态表示
2. 事件关系构建
3. 规则整合

行为图构建

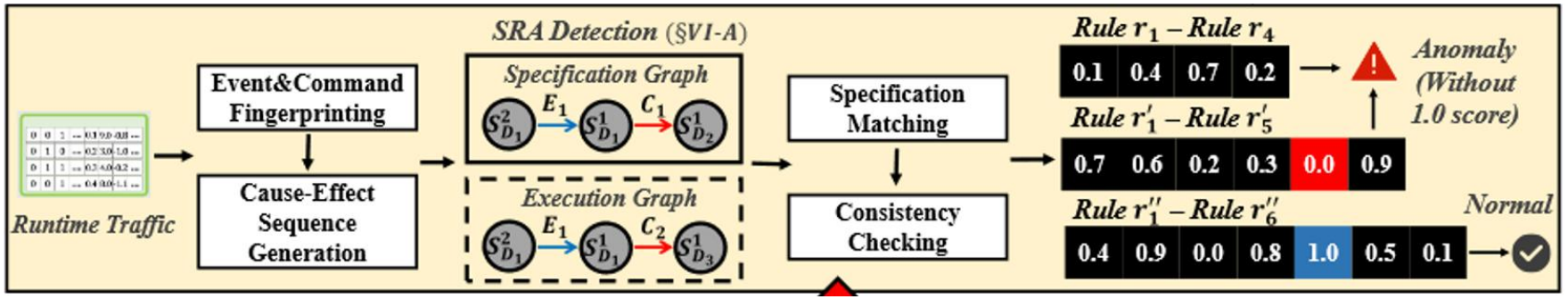


1. **设备状态表示**——使用设备的capability-value对，如门锁：{“contact”: “open”, “acceleration”: “active”}

   问题：复杂度为指数级，空间过大

   解决：使用其中一个capability的所有值
2. **事件关系构建**——设备状态节点连线
3. **规则整合**——构造Union节点来表示满足所有前提条件的中间状态。

SRA检测



工作流程

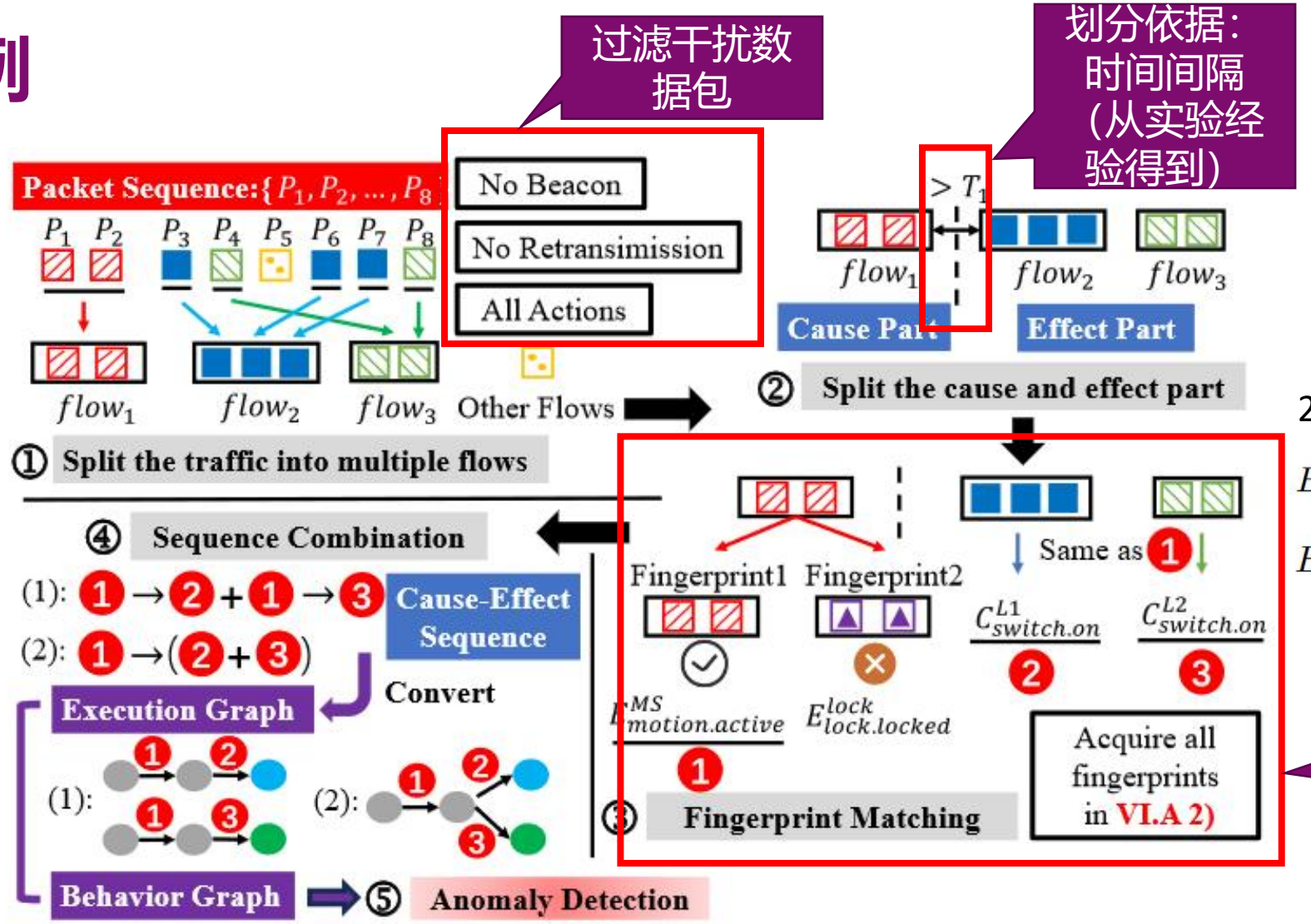
1. 捕获运行时流量
2. 指纹提取
3. 构建Cause-Effect序列
4. 序列匹配
5. 一致性检查

选该角度原因

开发人权限有限，仅凭设备状态信息无法确定自动化是否正确执行

检测举例

3个设备：
Aqara Motion Sensor (MS),
Mi Desk Lamp (L1),
Philipus Bedroom Lamp (L2)



2条规则：

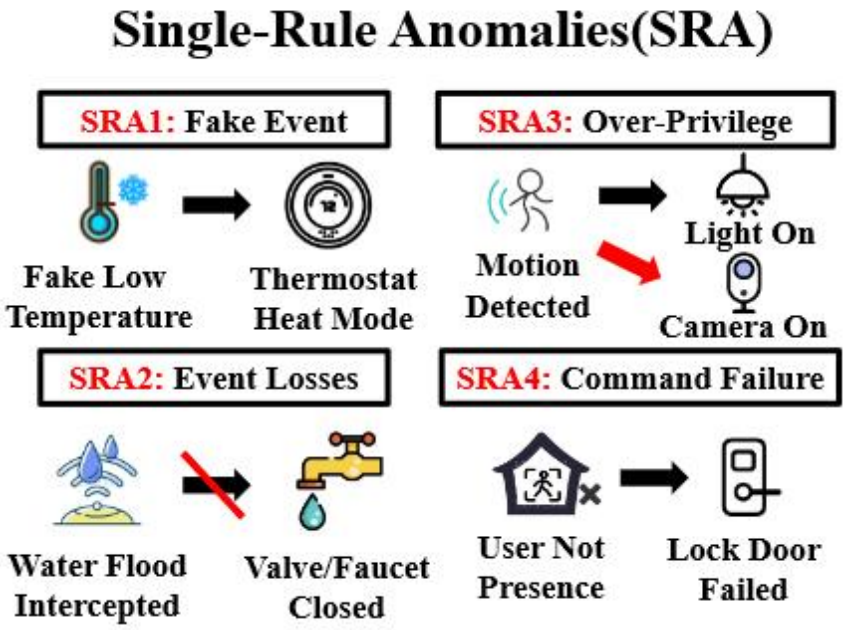
$$E^{MS}_{motion.active} \rightarrow C^{L_1}_{switch.on}$$

$$E^{MS}_{motion.active} \rightarrow C^{L_2}_{switch.on}$$

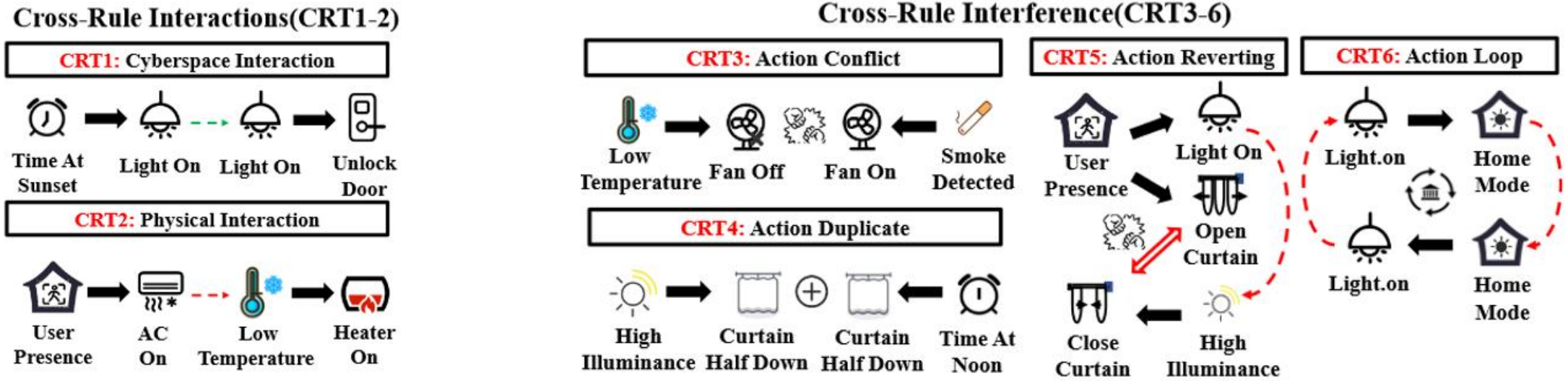
Match依据：计算曼哈顿距离

异常检查

1. Specification Matching
   - 将cause-effect sequence划分成**Sc**和**Se**两个集合
   - 对集合中的每个元素，在行为图中找到包含其的规则
   - 聚合规则，形成**specification graph**
2. Consistency Checking
   - 使用之前cause-effect sequence形成的**execution graph**判断
   - 使用GraphSAGE学习两个graph的特征
   - 使用余弦相似度来计算两个嵌入之间的相似度，以及事件部分Se和命令部分Sc的相似度
   - 若不存在精准匹配，存在异常

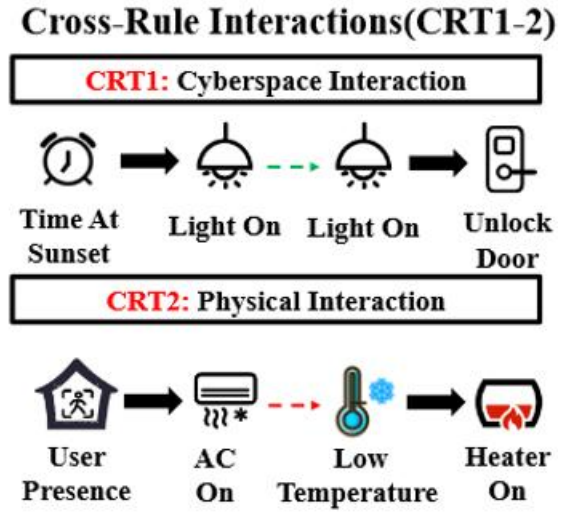


CRT检测



- 首先推导两个自动化规则(CRT1-2)之间的交互威胁。
- 在此基础上，在推导图上定位交叉规则干扰(CRT3-6) 。

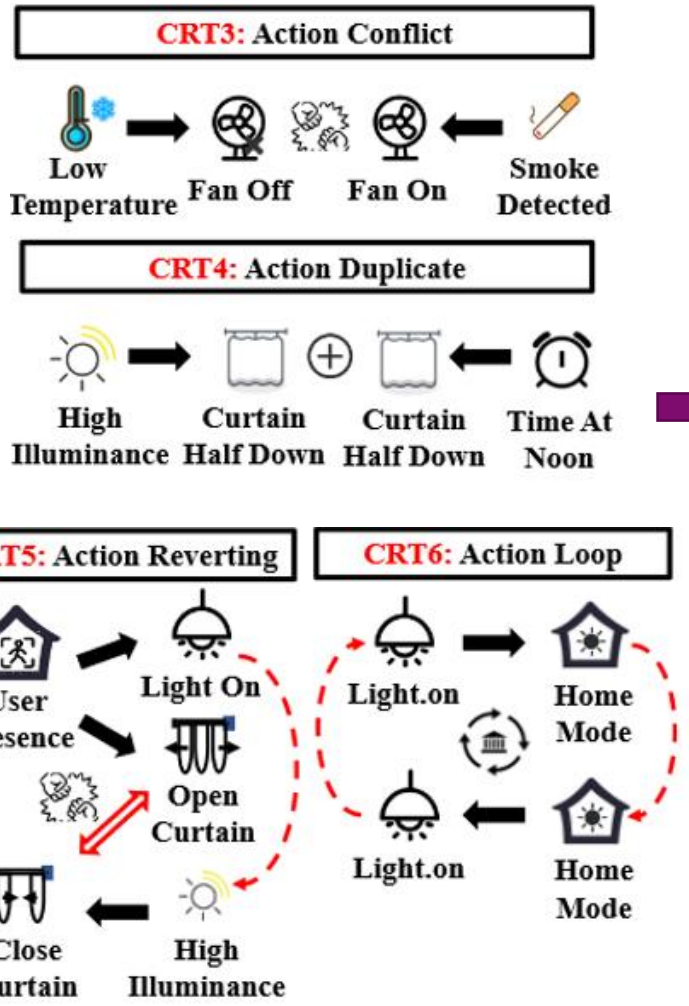
跨规则交互推理



如：(R1 : Er1 → Cr1) → (R2 : Er2 → Cr2).

- 网络空间的交互：规则R1的命令直接触发规则R2，因此规则R1的命令集Cr1必须包含规则R2的事件集Er2
- 物理通道交互：规则R1的命令集Cr1中的某些命令改变了物理环境，间接触发了规则R2。（文章划分了11种物理通道，利用BERT模型，将capability-value与物理通道匹配）

交叉规则干扰辨识

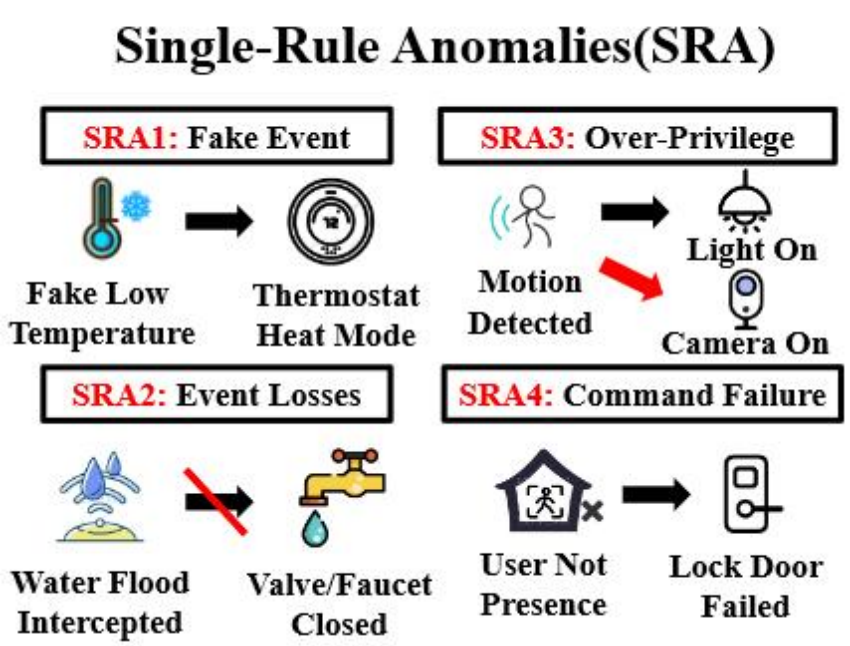


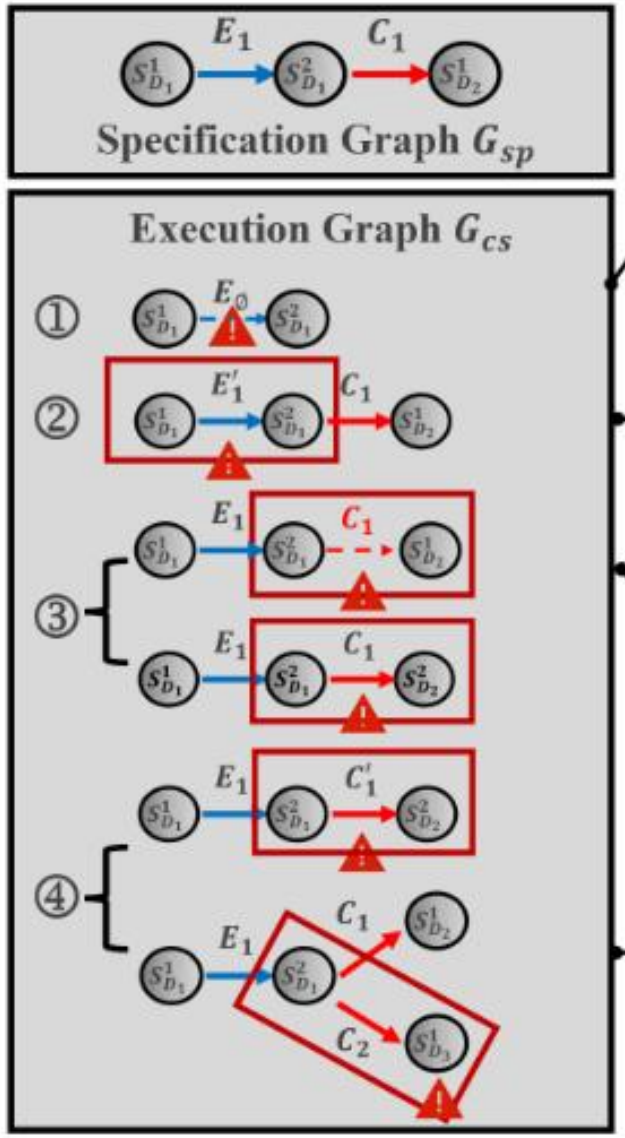
转为符号语言描述

Type	Representation
Action Conflict CRT3	$\exists q_1 \in C_{r1}, q_2 \in C_{r2}$ $S_1^s = q_1.suc, S_2^s = q_2.suc$ $s.t.\ \ S_1^s.cp = S_2^s.cp, S_1^s.val \neq S_2^s.val$
Action Duplicate CRT4	$\exists q_1 \in C_{r1}, q_2 \in C_{r2}$ $S_1^s = q_1.suc, S_2^s = q_2.suc$ $s.t.\ \ S_1^s.cp = S_2^s.cp, S_1^s.val = S_2^s.val$
Action Reverting CRT5	$\exists q_1 \in C_{r1}, q_n \in C_{rn}$ $S_1^s = q_1.suc, S_n^s = q_n.suc$ $s.t.\ \ S_1^s.cp = S_2^s.cp, S_1^s.val \neq S_n^s.val,$ $\forall_{i=1}^{n-1}(r_i, r_{i+1}) \in S_{cyb}/S_{phy}$
Action Loop CRT6	$s.t.\ \ E_{r1} \subseteq C_{rn}$ $\forall_{i=1}^{n-1}(r_i, r_{i+1}) \in S_{cyb}/S_{phy}$

- 将这些表示转换为约束，在图上找到满足这些表示的任何规则组。
- 对于CRT3-4，返回两个匹配规则(r1, r2)和冲突命令(Cr1, Cr2)或公共命令部分(Ccom)。
- 对于CRT5-6，返回匹配规则链(r1, r2，…, rn)

威胁缓解——SRA





Interpretation

① **Anomaly:** SRA2
Cause: Network Delay/Attacker
Solution: Checking network and put device D_1 closer to the router/gateway.

② **Anomaly:** SRA1
Cause: Device Defect/Interference/Attacker
Solution: **(1)** Update firmware.
(2) Design device start and stop strategy.

③ **Anomaly:** SRA3
Cause: Device Defect/ Network Delay /Attacker
Solution: **(1)** Update firmware.
(2) Checking network and put device D_2 closer to the router/gateway.

④ **Anomaly:** SRA4
Cause: App Defect /Attacker
Solution: Set limited device management permissions for rules

威胁缓解——CRT

- **CRT1-2**
  - 根本原因是用户没有意识到意外触发的规则和意外的设备行为
  - 向用户发送关于所有触发规则的消息，并通过确定意外触发规则发送的命令是否包含危险操作来附加风险级别(高/低)

- **CRT3-6**
  - 动作冲突和动作重复(CRT3-4)：为每个规则设置优先级，并阻止低优先级规则的执行。优先级顺序:“用户活动>物理相关>物理无关>模式相关>时间相关”
  - 动作恢复(CRT5)：添加上下文状态条件，只有当事件在最近的时间T内没有发生才执行该规则。T的设定由经验决定
  - 动作循环(CRT6)：记录执行次数，一定时间内未执行才会执行

04 实验效果与总结

自动化规则提取表现

TABLE IV: The accuracy of rule extraction on four platforms.

Platform	Word2Vec [47]		BERT [38]	
	DAnalysis	+CAnalysis	DAnalysis	+CAnalysis
SmartThings(105)	89.52%	92.38%	91.43%	97.14%
Apple Homekit(128)	89.06%	96.09%	92.97%	99.22%
Google Home(160)	83.13%	95.63%	91.25%	98.13%
Xiaomi Home(192)	85.94%	91.15%	88.02%	98.96%

不使用TF-IDF原因：认为描述中所有的内容都是重要的

SRA检测表现

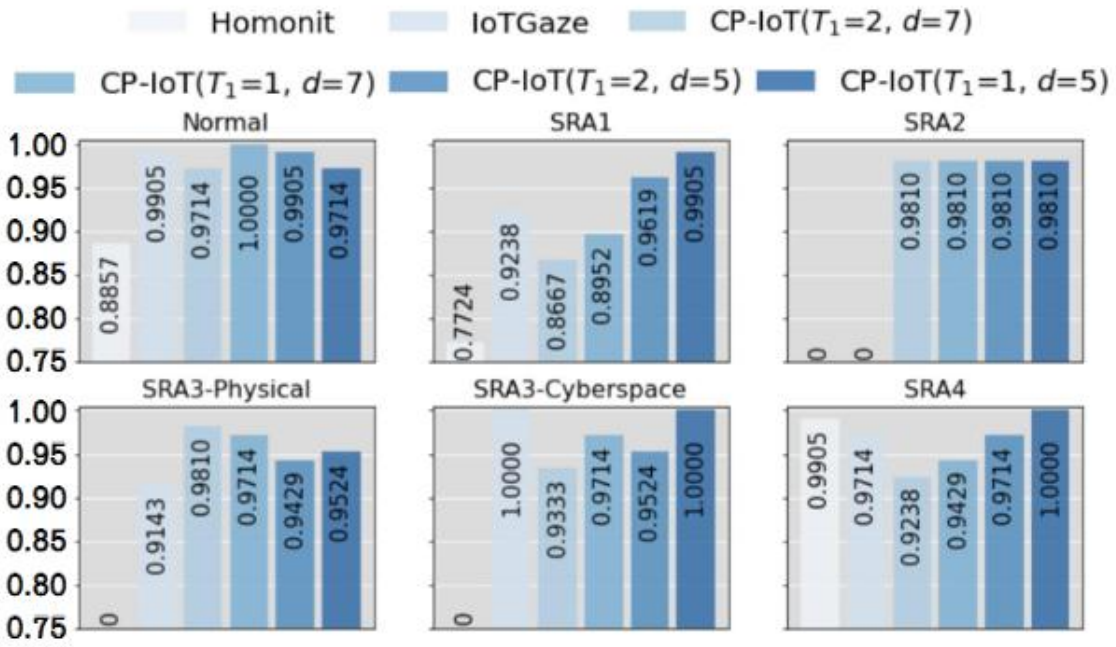


Fig. 11: SRA detection accuracy on SmartThings.

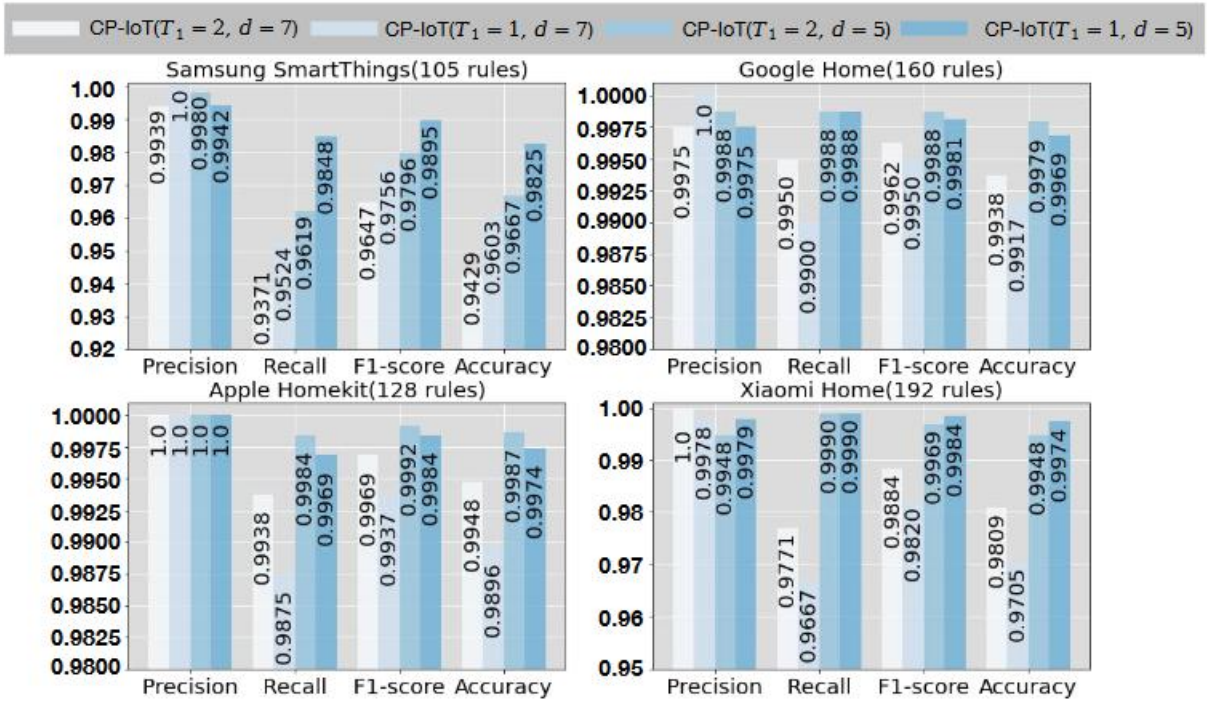


Fig. 14: SRA detection performance on four platforms.

CRT检测表现

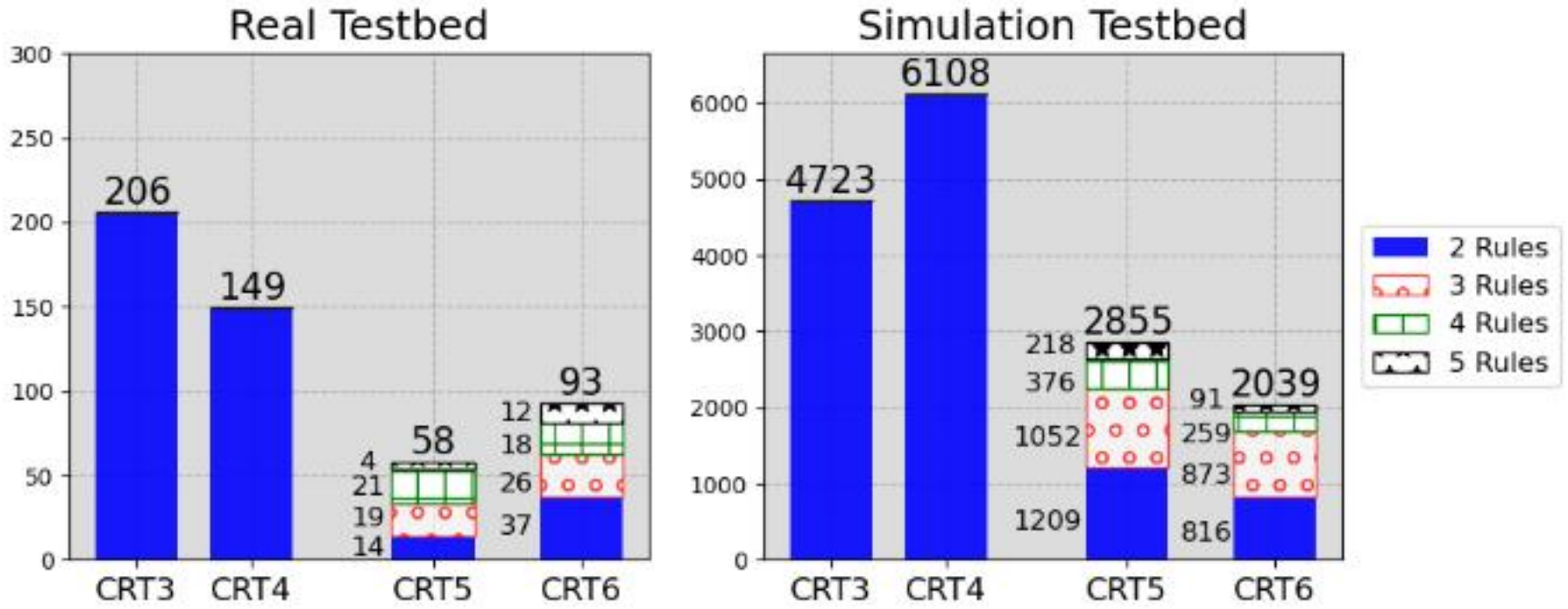


Fig. 15: The number of CRT3-6 discovered by CP-IoT.

总结

- 提出了一种智能家居跨平台监控系统CP-IoT，能够发现单平台和跨平台的威胁。
- 提出了一种基于应用页面的规则提取方法和一种多粒度（包级、流级）规则行为识别技术，可以解决平台之间的架构和流量差异。
- 构建了一个centralized图来描绘设备状态变化和跨平台规则执行。
- 设计了两种算法来发现各种规则执行异常和跨规则威胁，并提出了缓解这些威胁的方法。

方案缺陷

1. 当有经验的攻击者掌握了设备与平台之间的通信模式后，重放的事件报文不会被CP-IoT检测到，因为它不违反事件指纹(如MitM攻击)。
2. CRT Mining的空间复杂度为O(n2)，在大规模场景下耗费大量时间。
3. 检测事件丢失和一些命令失败情况需要通过获取日志来手动确定设备状态，这需要开发人员的许可
4. 配置方案分析涉及用户隐私

改进思考

- 多平台控制通道规则是否能够检测？
- 能否加上平台、应用状态辅助检测，防范中间人攻击？

2024.03.07

恳请批评指正

汇报人：付文轩