IoT Security: Smart Home Devices

Yue Duan

based on slides from

Outline

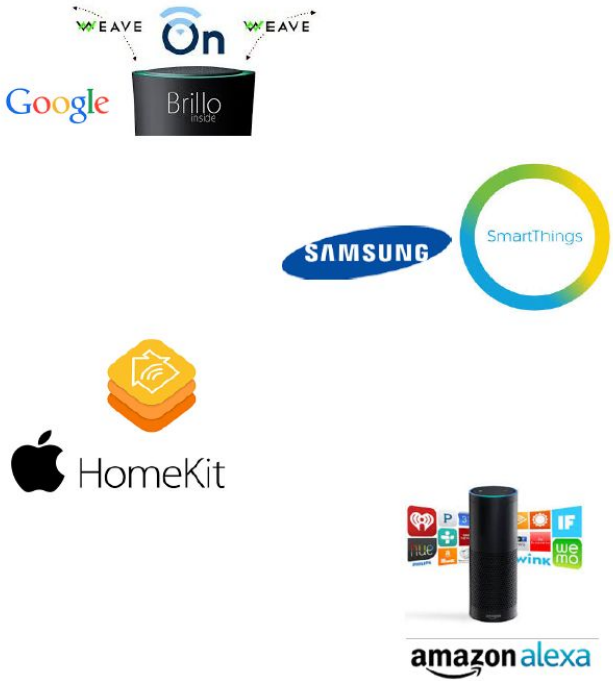
- Research paper:
  - Security Analysis of Emerging Smart Home Applications
  - SmartAuth: User-Centered Authorization for the Internet of Things
  - Sensitive Information Tracking in Commodity IoT

Security Analysis of Emerging Smart Home Applications

Earlence Fernandes, Jaeyeon Jung, Atul Prakash

IEEE S&P 2016

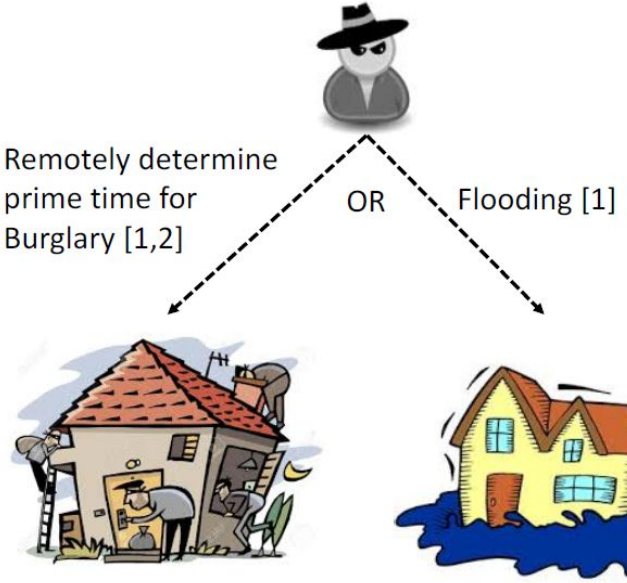
Motivation



Emerging Smart Home Frameworks

Motivation

Potential Security Risks



[1] Denning et al., Computer Security and the Modern Home, CACM'13
[2] FTC Internet of Things Report'15

Current Vulnerabilities

These attacks are device-specific, and require proximity to the home

Motivation

- In what ways are these emerging, programmable smart homes vulnerable to attacks?
- What do those attack entail?

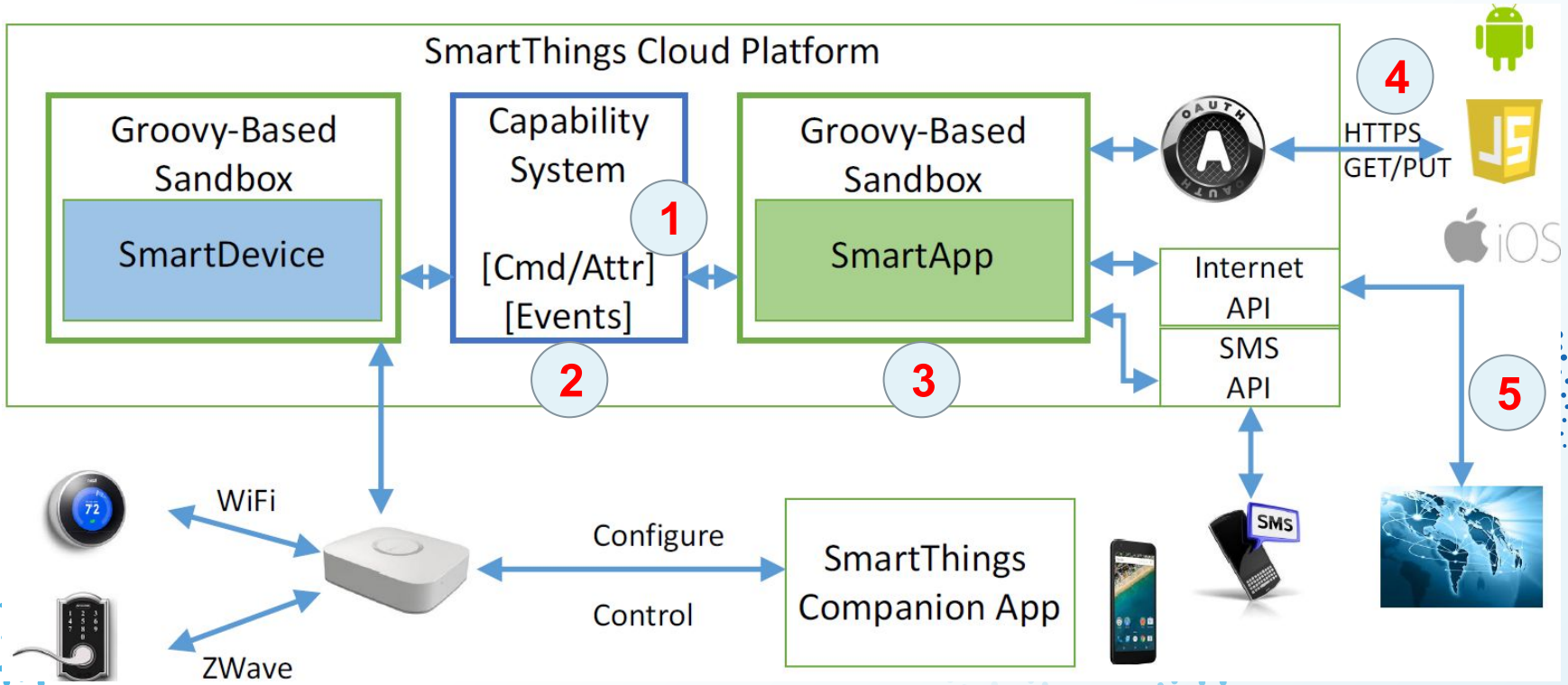
Motivation

- Why SmartThings?
  - relatively mature
  - 521 smartApps
  - 132 device types
  - share design principles with other existing nascent frameworks
- Methodology
  - examine security from **5 perspectives** by constructing test apps to exercise SmartThings API
  - empirical analysis of **499 apps** to determine security issue prevalence
  - **proof of concept attacks** that compose security flaws

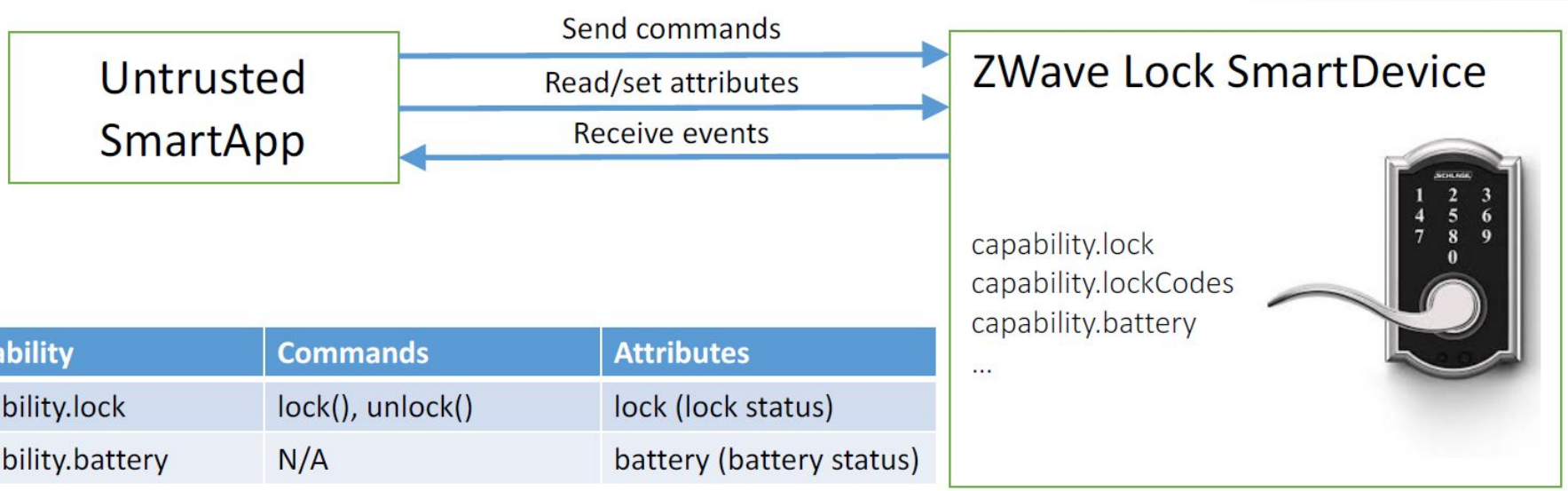
Results Overview

Security Analysis Area	Finding
Overprivilege in Apps	Two Types of Automatic Overprivilege
Event System Security	Event Snooping and Spoofing
Third-party Integration Safety	Incorrect OAuth Can Lead to Attacks
External Input Sanitization	Groovy Command Injection Attacks
API Access Control	No Access Control around SMS/Internet API
Empirical Analysis of 499 Apps	**> 40% of apps exhibit overprivilege of atleast one type**
Proof of Concept Attacks	**Pincode Injection and Snooping, Disabling Vacation Mode, Fake Fire Alarms**

SmartThings Primer



Capability System



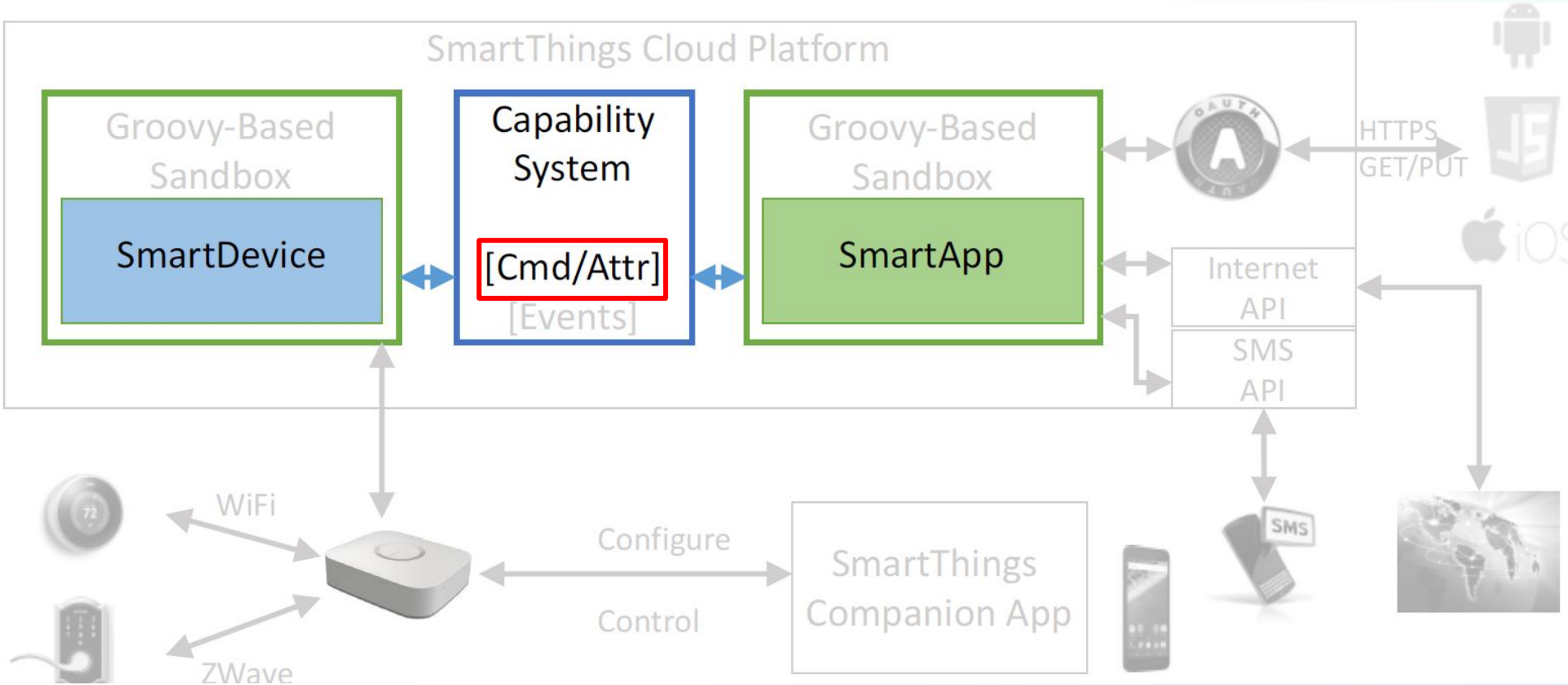
Capability	Commands	Attributes
capability.lock	lock(), unlock()	lock (lock status)
capability.battery	N/A	battery (battery status)

Usability
Simpler Coarser Capabilities

Ease of Development
Expressive Functionality

Security
Very Granular Capabilities

Overprivilege in SmartApps

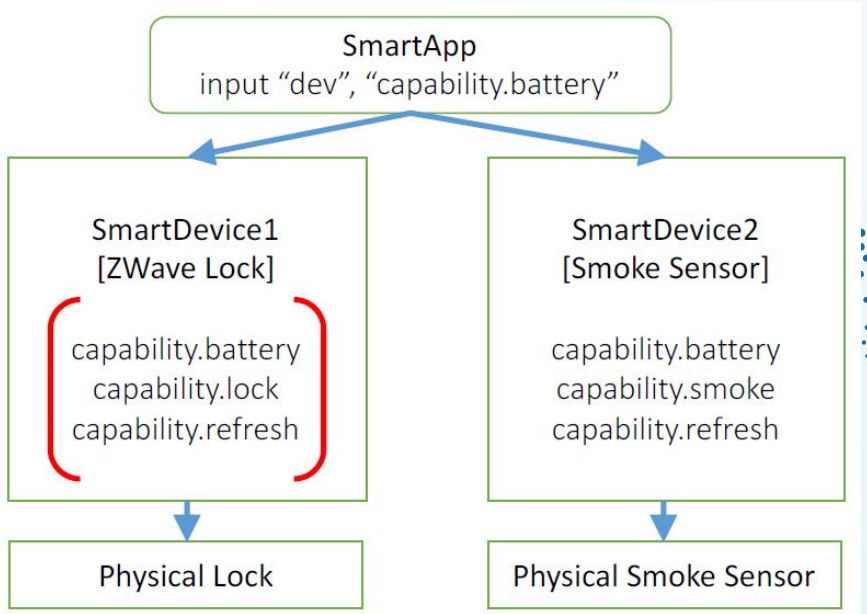


Overprivilege in SmartApps

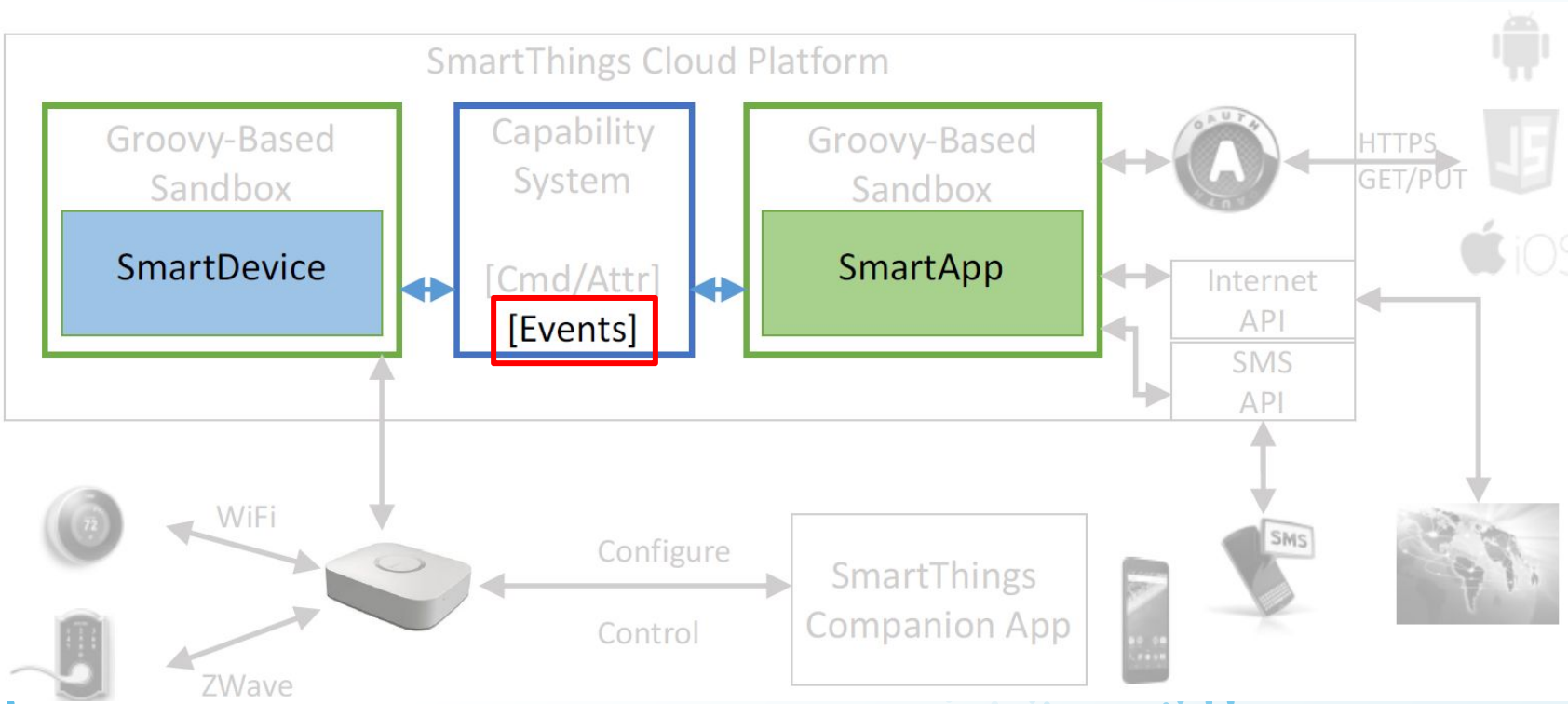
- Coarse-grained Capabilities
  - ‘auto-lock’ app from app store
  - only needs ‘lock’ command, but can also issue ‘unlock’

Overprivilege increases attack surface of the Home

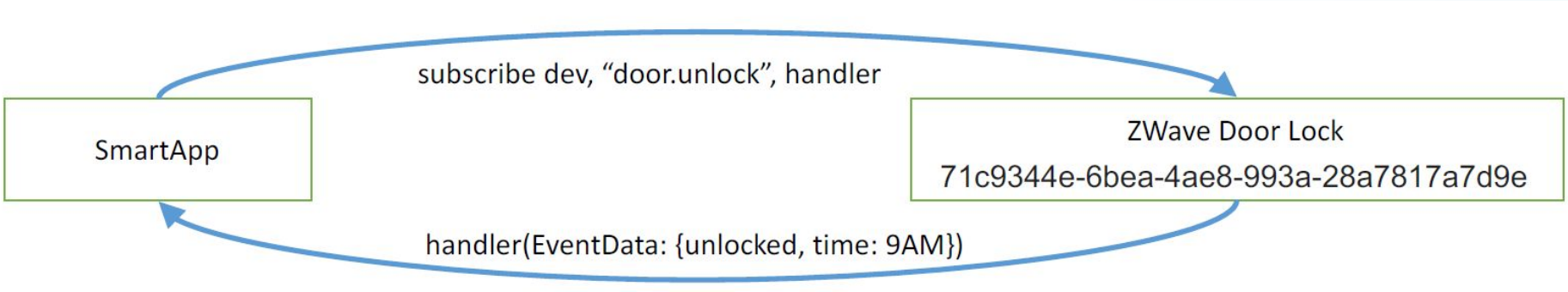
Coarse SmartApp-SmartDevice Binding



Insufficient Event Data Protection

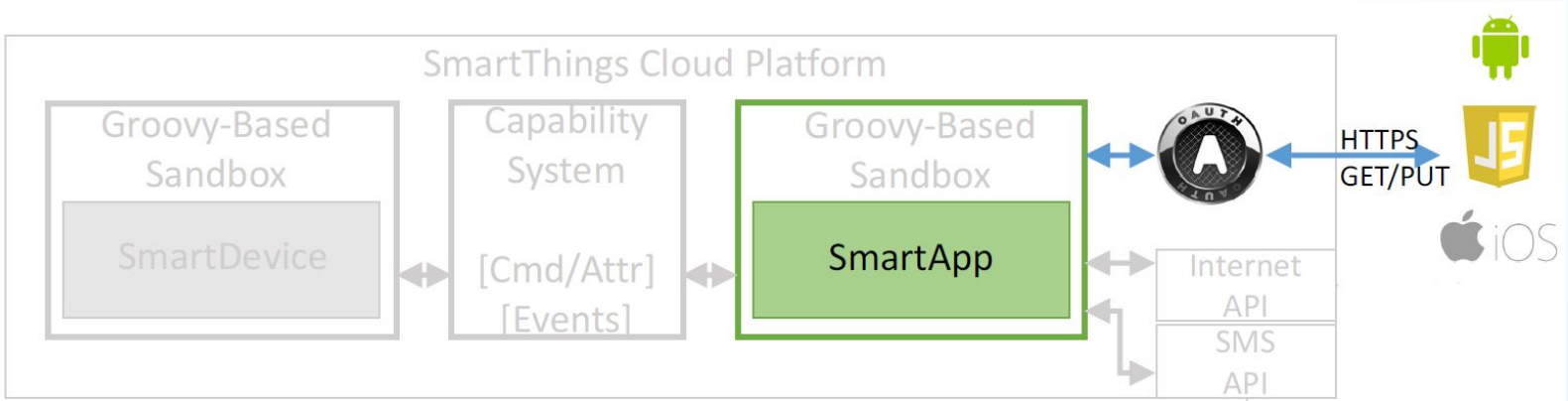


Insufficient Event Data Protection



- Once a SmartApp gains **any** capability for a device, it can subscribe to **any event** that device generates
- If a SmartApp **acquires the 128-bit ID**, then it can monitor all events of that device **without** gaining any of the capabilities the device supports
- Using the 128-bit ID, a SmartApp can **spoof physical device events**

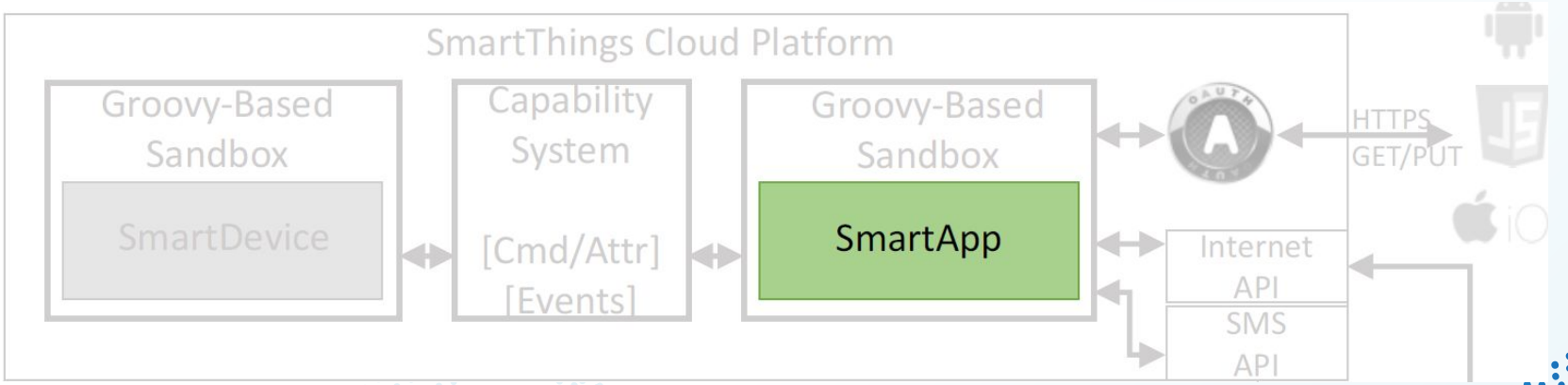
Other Potential Security Issues - OAuth



- Insecurity of Third-Party Integration:
  - SmartApps expose HTTP endpoints protected by OAuth;
- **Incorrect implementation** can lead to remote attacks [1]

[1] Chen et al., OAuth Demystified for Mobile Application Developers, CCS'14

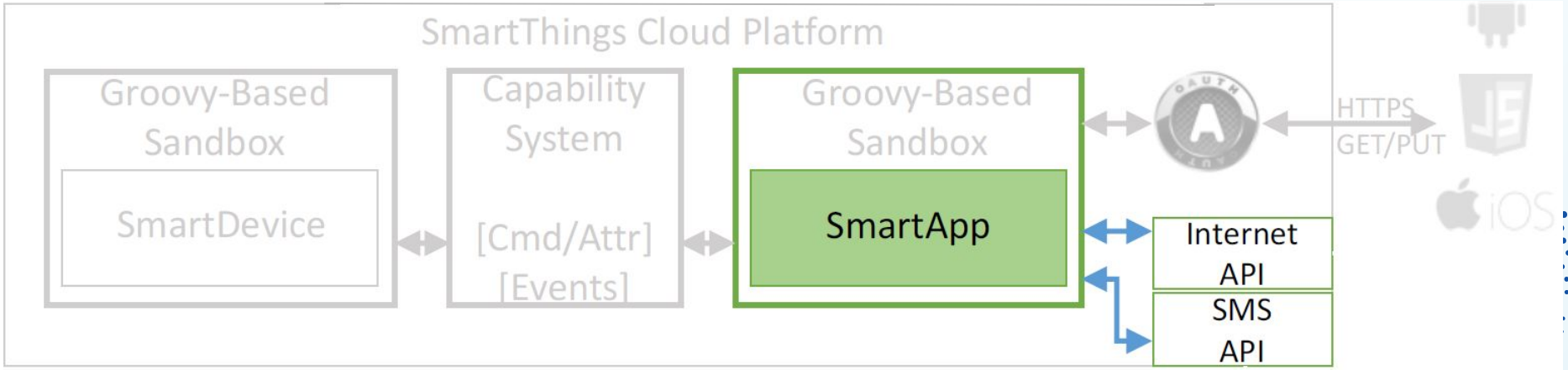
Other Potential Security Issues - Groovy



- Unsafe use of Groovy Dynamic Method Invocation:
  - Apps can be tricked into performing **unintended actions**

```
def foo() { ... }
def str = "foo"
"$str"()
```

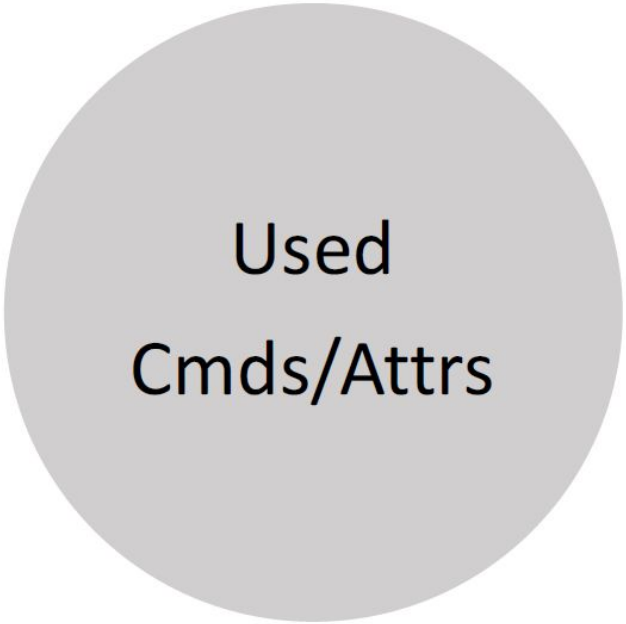
Other Potential Security Issues – Unrestricted External Communication APIs



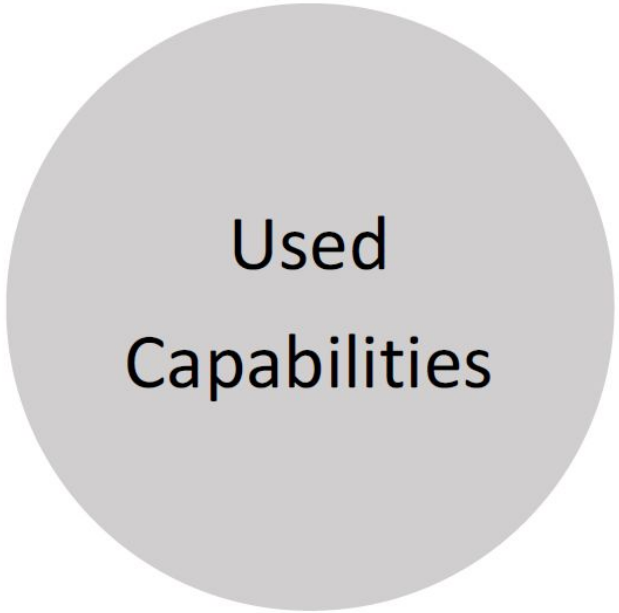
- Unrestricted Communication Abilities:
  - SMS and Internet
- Can be used to **leak data arbitrarily**

Computing Overprivilege

Coarse-Grained Capabilities



Coarse SmartApp-SmartDevice Binding



Measuring Overprivilege in SmartApps

Challenges

- Incomplete capability details (commands/attributes)
- SmartThings is closed source; can’t do instrumentation
- Groovy is extremely dynamic; Bytecode uses reflection (Groovy Meta Object Protocol)

Solutions

- Discovered an unpublished REST endpoint, which, if given a device ID, returns capability details
- Study source code of apps from open-source app store instead
- Static analysis on AST

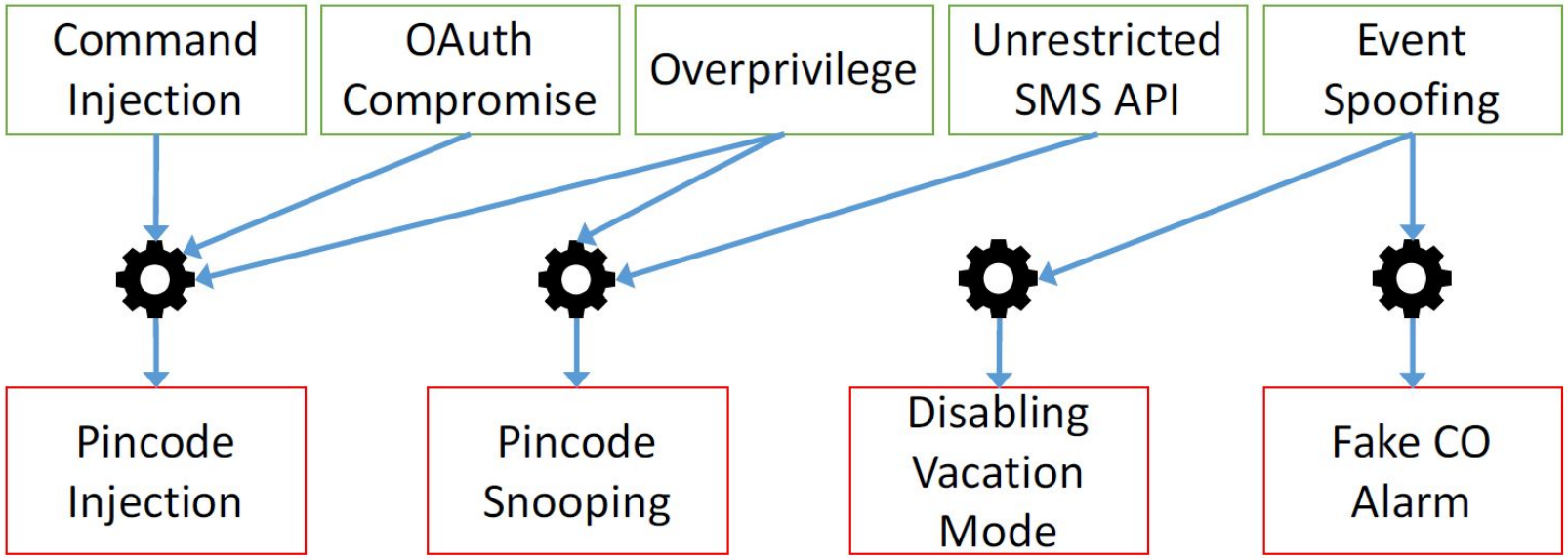
Empirical Analysis Results

	Documented	Completed
Commands	65	93
Attributes	60	85

Reason for Overprivilege	Number of Apps
Coarse-grained Capability	276 (55%)
Coarse SmartApp-SmartDevice Binding	213 (43%)

Overprivilege Usage Prevalence (Coarse Binding)	68 (14%)

Exploiting Design Flaws in SmartThings



Popular Existing SmartApp with Android companion app; Unintended action of setCode() on lock

Stealthy malware SmartApp; ONLY requests capability.battery

Malware SmartApps with no capabilities; Misuses logic of existing SmartApps with fake events

Potential Defense Strategies

- Achieving least-privilege in SmartApps
  - **Risk asymmetry** in device operations, e.g., oven.on and oven.off
  - Include notions of risk from multiple stakeholders, rank [1], and regroup

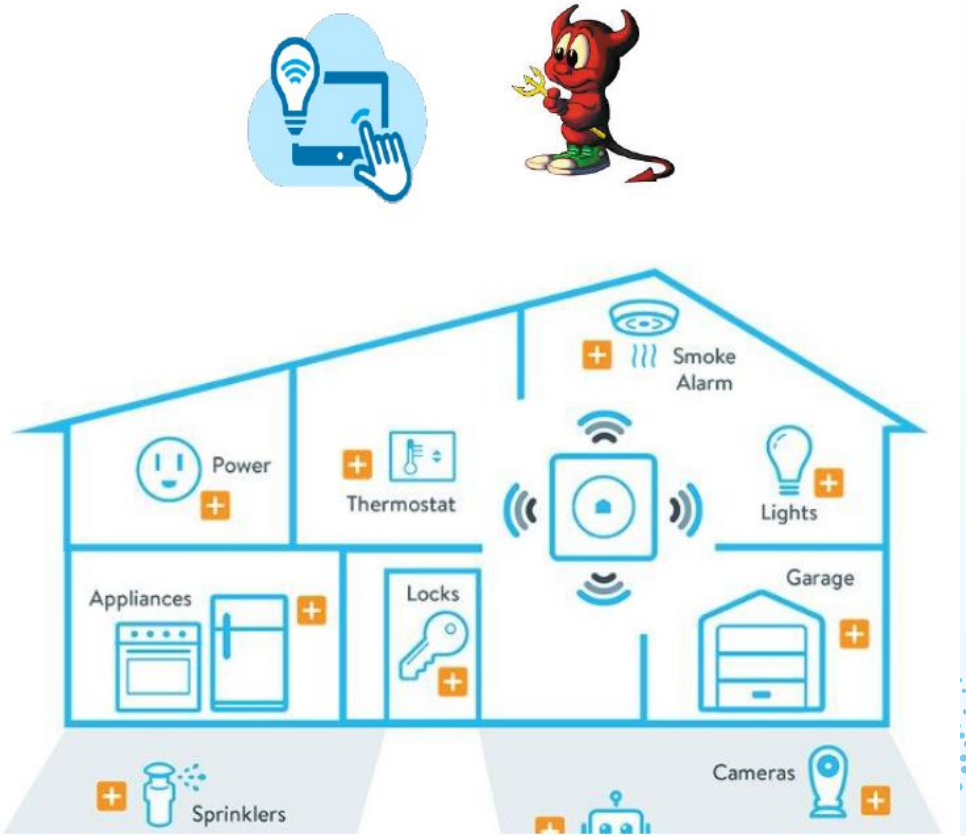
- Preventing information leakage from events
  - Provide a notion of **strong identity** for apps + **access control on events**
  - Make apps request access to certain types of events, e.g., lock pincode ACKs

SmartAuth: User-Centered Authorization for the Internet of Things

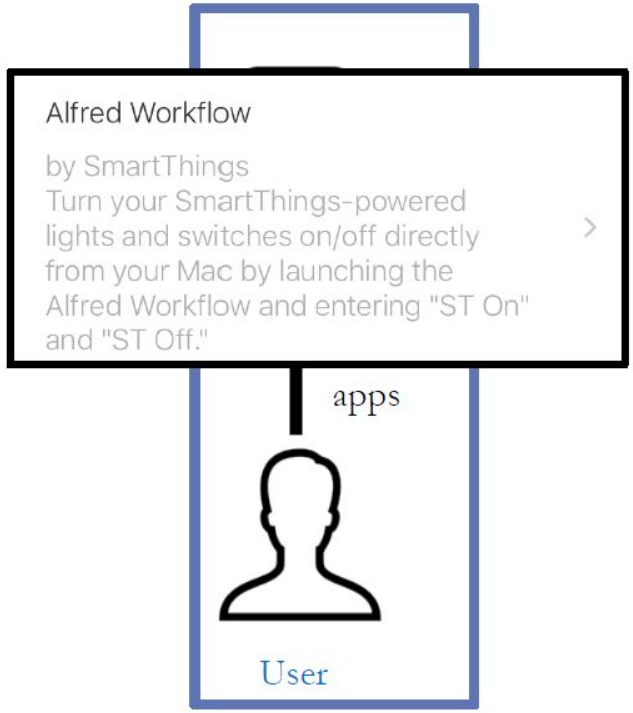
Yuan Tian, Nan Zhang, Yueh-Hsun Lin, XiaoFeng Wang, Blase Ur, XianZheng Guo and Patrick Tague

USENIX SEC 2017

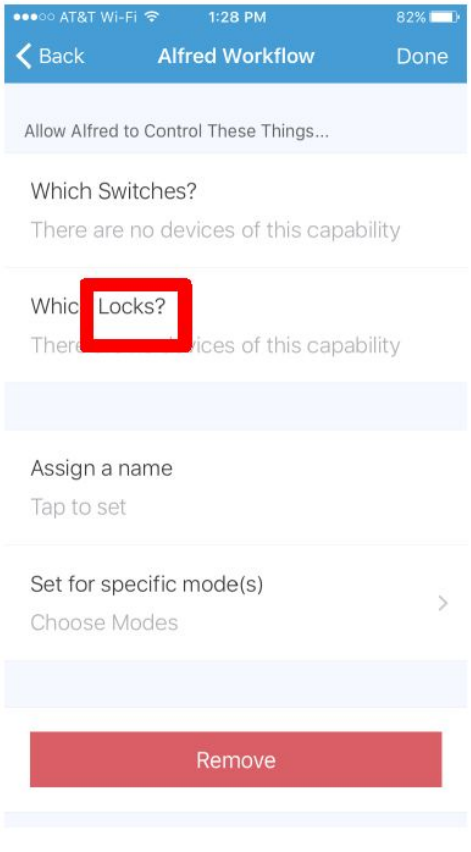
Smart-home apps improve quality of life, but can be **risky**



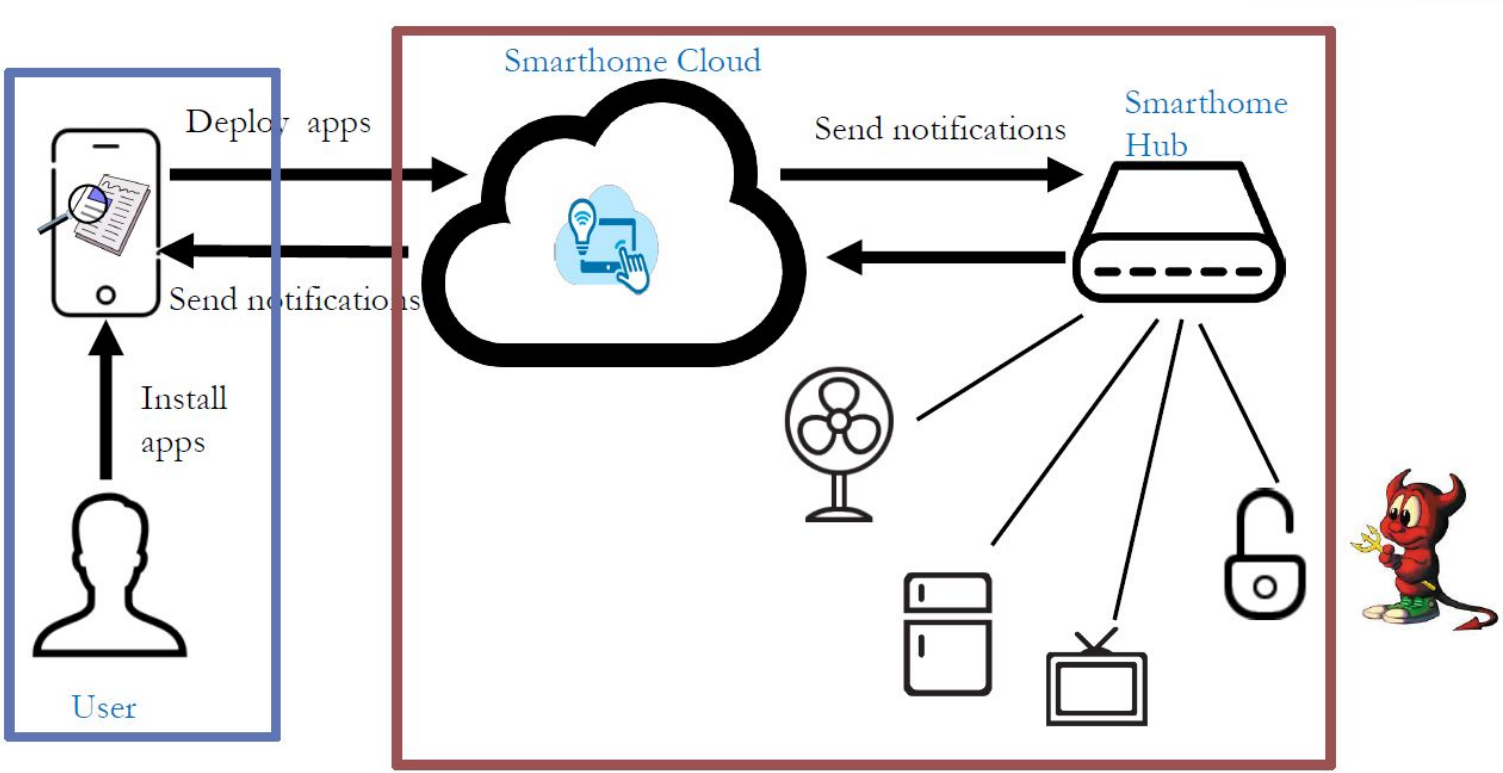
Users have limited information about what is going on



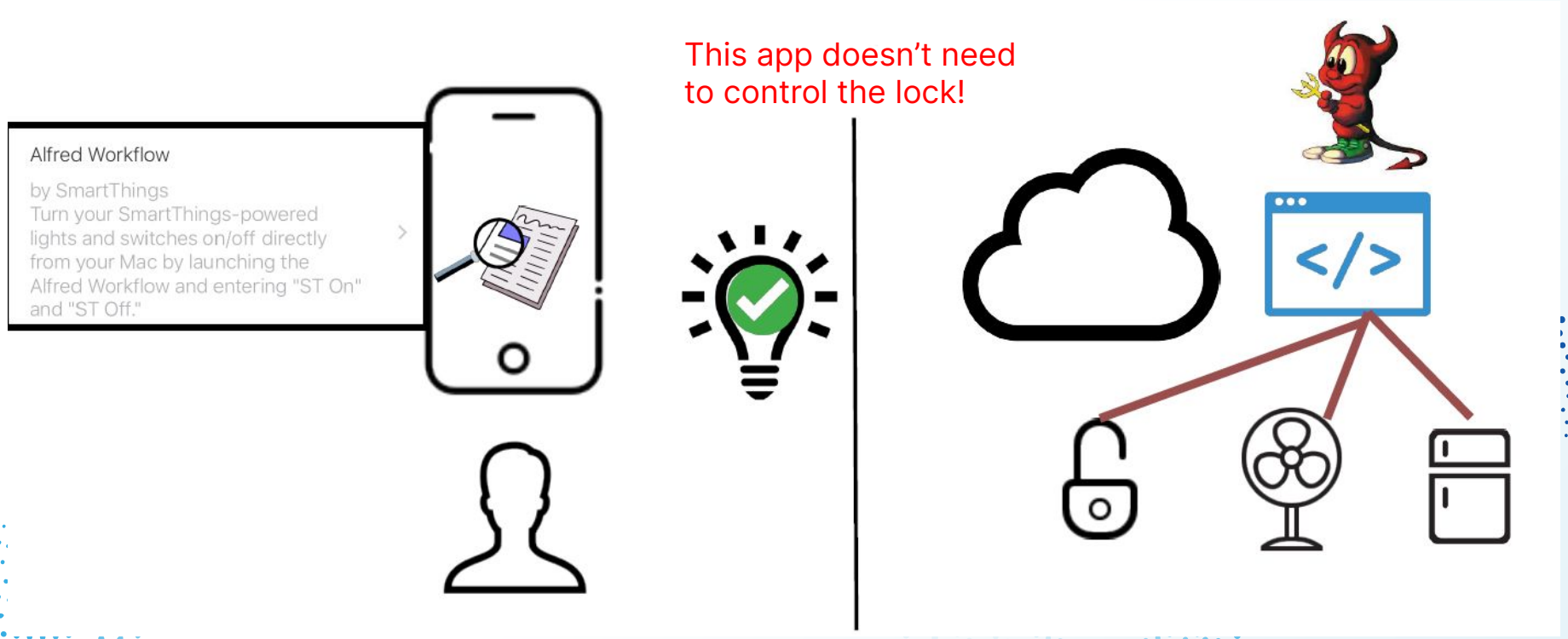
Functionalities explained to the user



Users have limited information about what is going on



Can we notify users about the most important information?



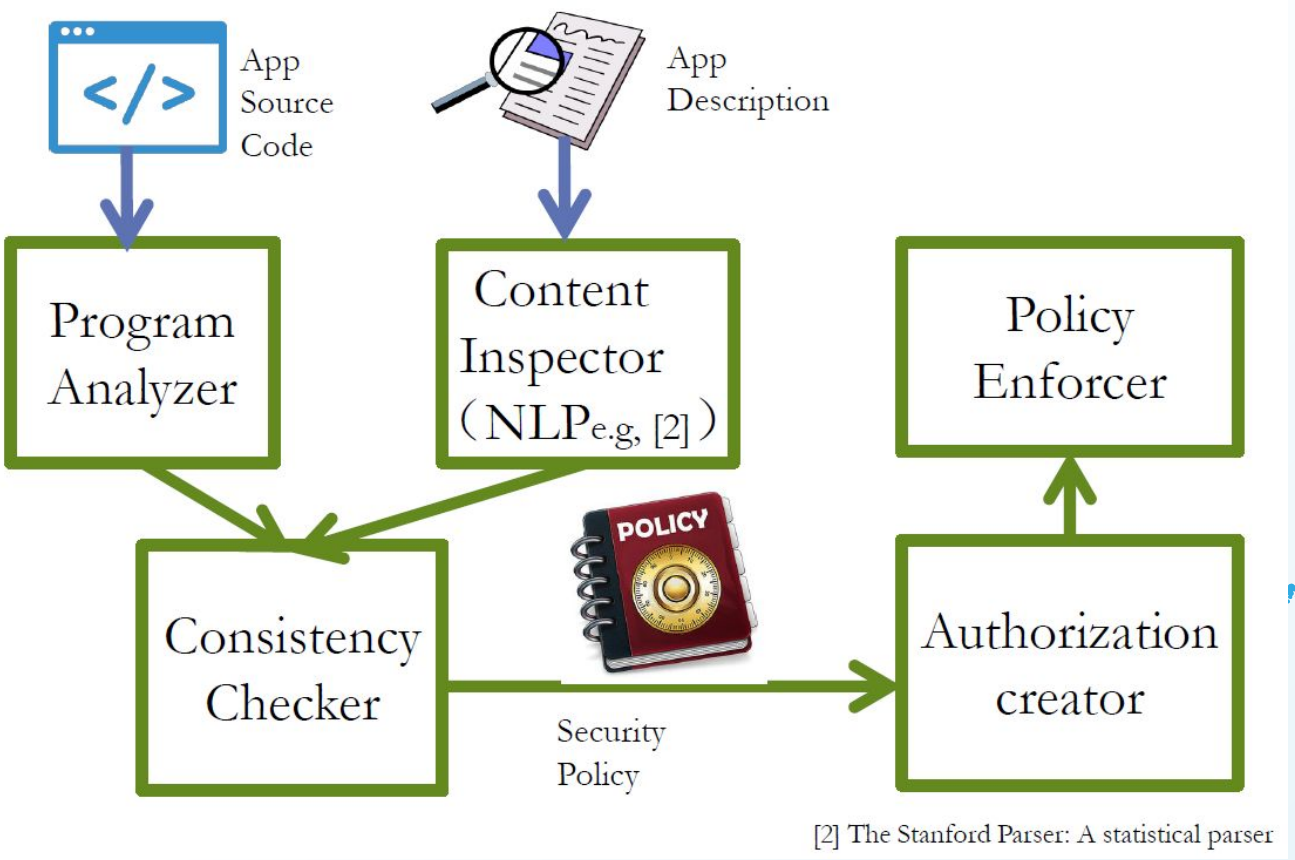
Challenges

- Security and privacy implications depend on **context**
  - Same sensor in bedroom vs. outside has very different implications
- Behaviors in code cannot be mapped directly to high-level functionality in description
- Need to support cross-device scenarios

Redesign the authorization system

- Goals:
  - Security and Privacy:
    - Share minimum data and capabilities for desired functionality
  - IoT specific:
    - Cross-device, context-based, automatic control
  - Usability:
    - Assist user to make well-informed decisions, minimize user burdens
  - Performance:
    - Lightweight and compatible

SmartAuth Overview

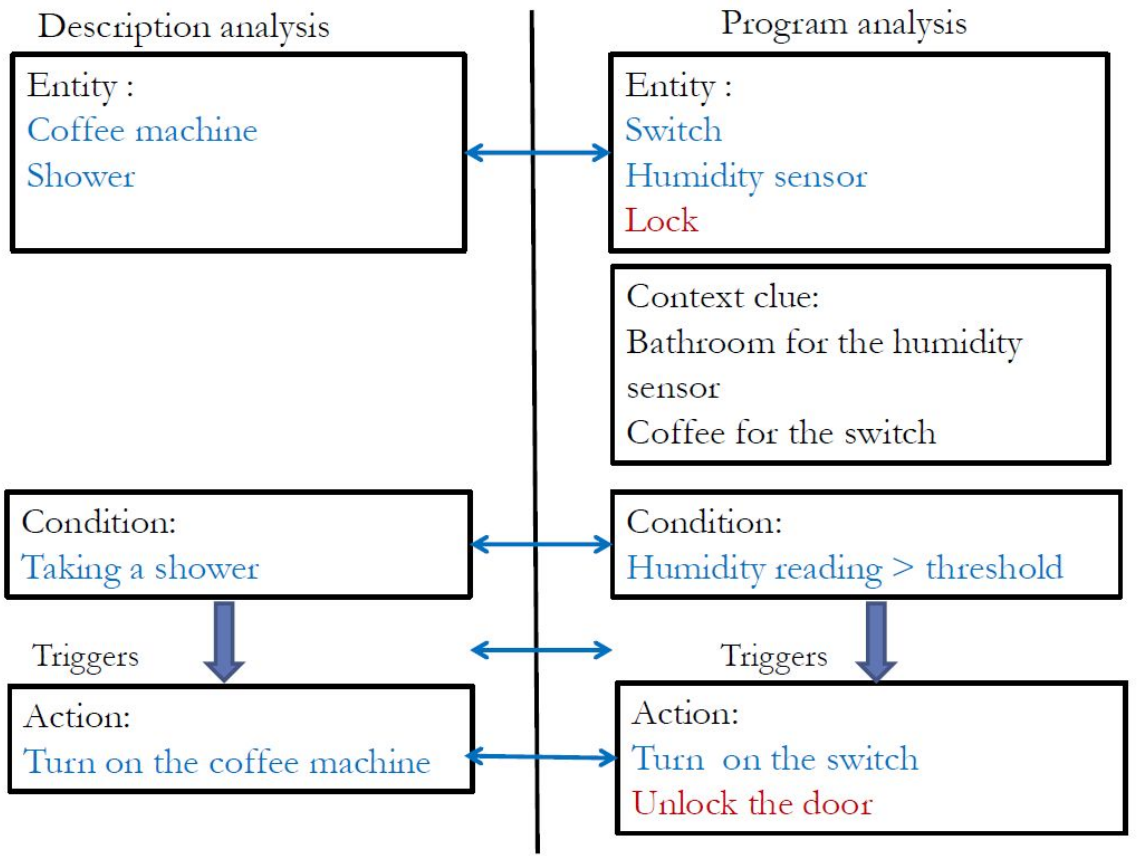


[2] The Stanford Parser: A statistical parser

Example - Program Analyzer

```
section("Bathroom humidity sensor") {
        input "bathroom", "capability.relativeHumidityMeasurement",
        title: "Which humidity sensor?"
    }


 if (shower.value.toInteger() > relHum) {
            coffee.on()
```

Example - NLP and Behavior Correlation



Evaluation

- How effective is SmartAuth?
  - How accurate is the policy extraction?
  - How does SmartAuth impact users' decisions?
- What is the performance overhead?
- How compatible is SmartAuth?

Evaluation: Effectiveness of extracting policies

- Manual analysis to verify all the cases
- 3.9% false positives
  - Limitations of NLP analysis
- No false negatives

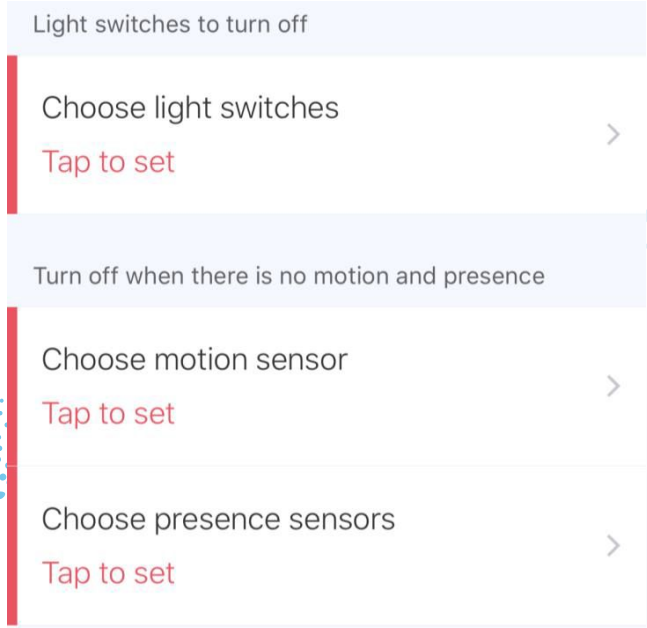
Evaluation: User study

- Between-subjects, in-lab study
- 100 participants split into two groups:
  - SmartAuth
  - Current SmartThings interface (manifest-style)
- Five pairs of similar apps
  - Participant chooses one of the two
  - One has unexpected privileges

Example app pairs

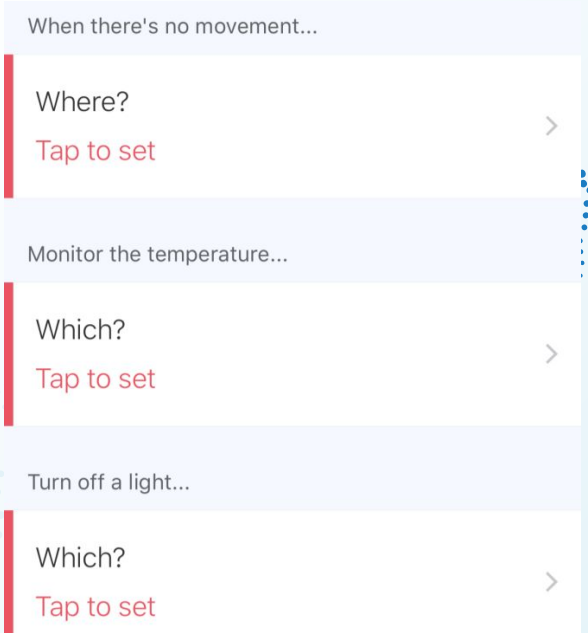
Lights Off

- Turn lights off when no motion or presence detected for a period of time

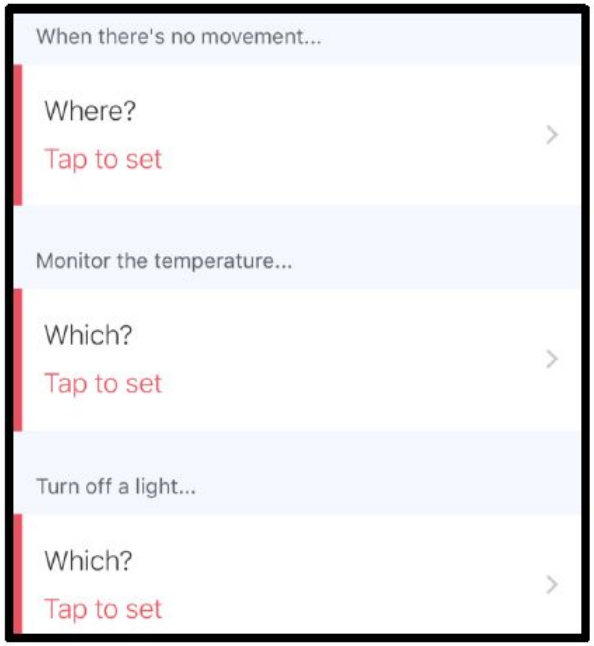


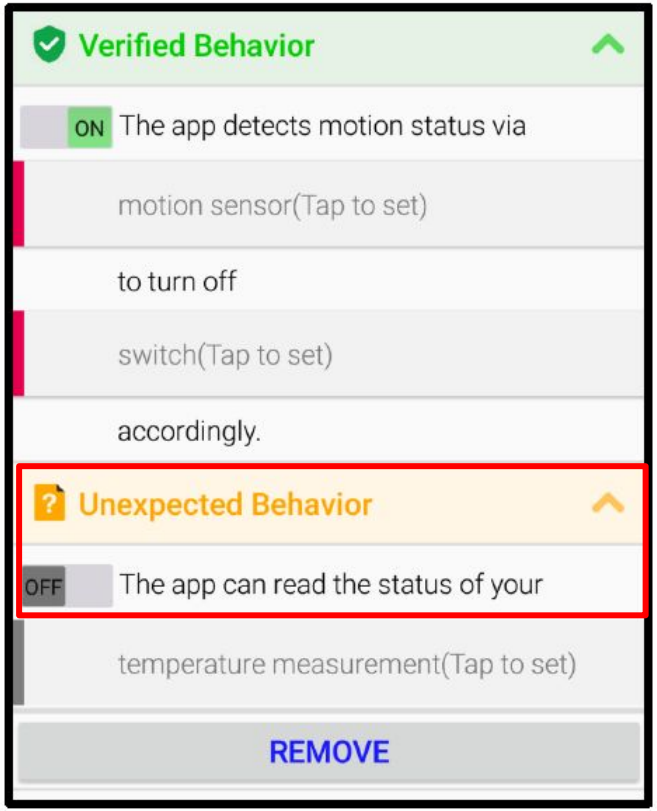
Darken Behind Me

- Turn your lights off after a period of no motion being observed

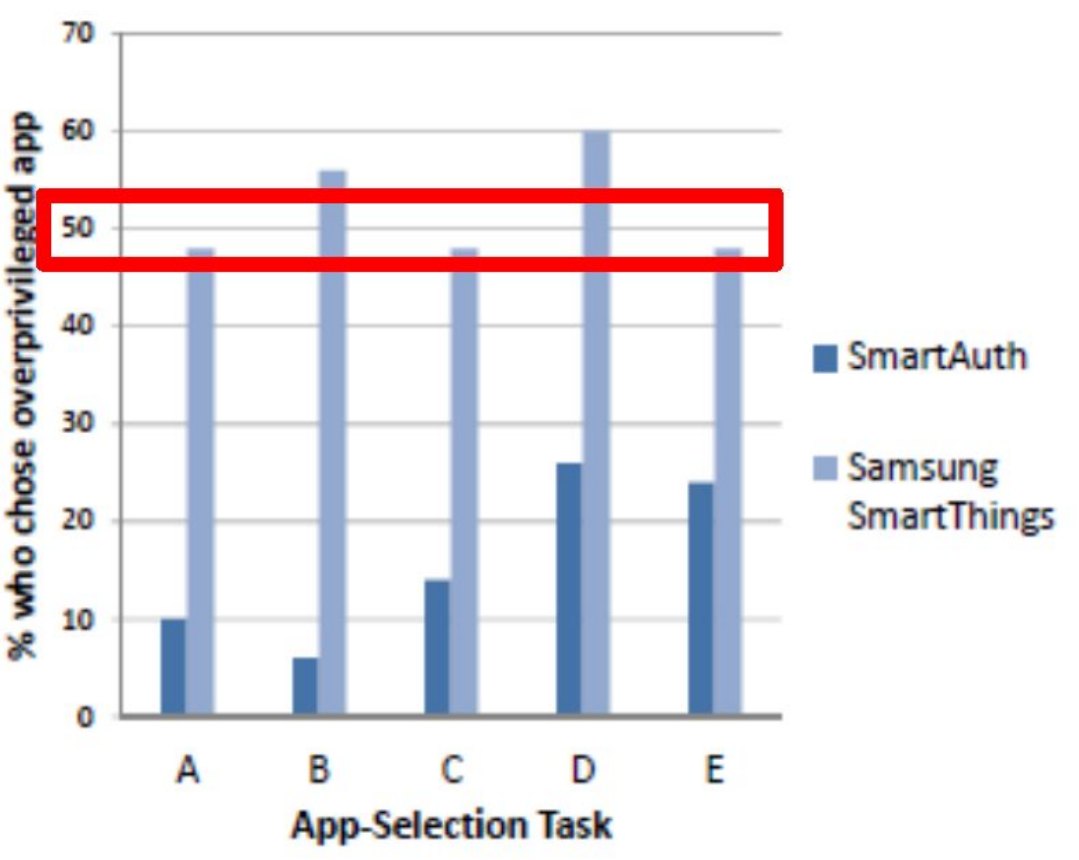


SmartThings VS SmartAuth



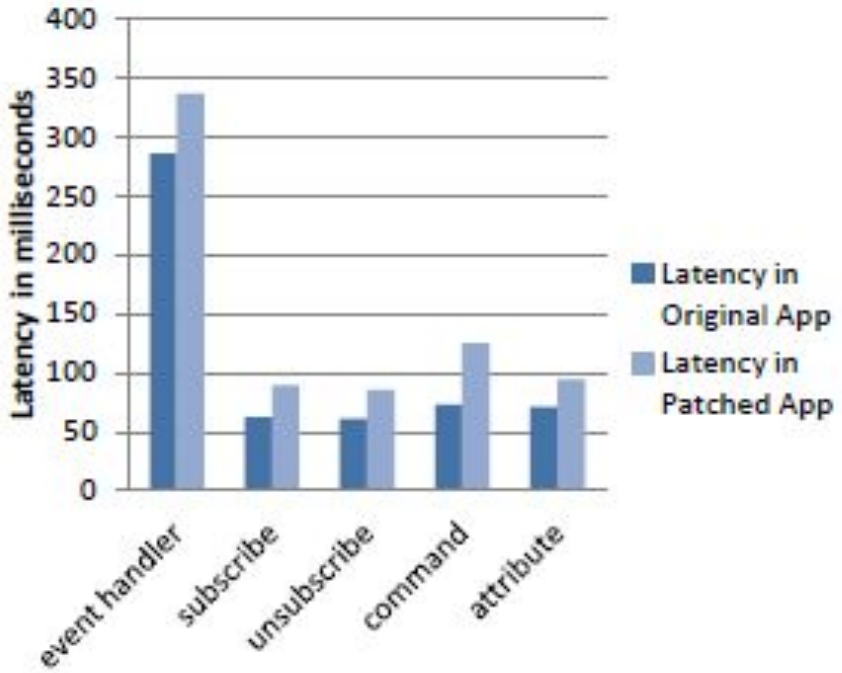


Users make better decisions with SmartAuth



Evaluation: Performance

- Run-time enforcement



Evaluation: Compatibility

- Observe behaviors and debug information
- None of the apps crash
- In the extreme case, 3.3% of apps loss functionality when we block all remote access

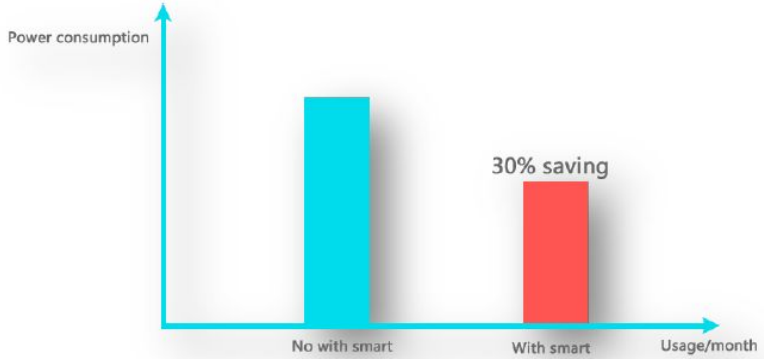
Sensitive Information Tracking in Commodity IoT

Z. Berkay Celik, Leonardo Babun, Amit K. Sikder, Hidayet Aksu, Gang Tan, Patrick McDaniel, and Selcuk Uluagac

USENIX SEC 2018

Internet of Things (IoT) enables the future

Smart Homes



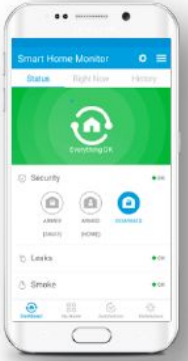
Smart Energy

Healthcare

Smart Farms

IoT is no magic

Connected devices



Mobile app

Automation

```
MQTT.sub(topicInLedA, function(conn, topic, msg) {
  print('Topic:', topic, 'message:', msg);
  if (msg === '0'){
    GPIO.write(pinLedA,0);
    isLedAOn = 0;
  } else {
    GPIO.write(pinLedA,1);
    isLedAOn = 1;
  }
}, null);


MQTT.sub(topicInLedB, function(conn, topic, msg) {
  print('Topic:', topic, 'message:', msg);
  if (msg === '0'){
    GPIO.write(pinLedB,0);
    isLedBOn = 0;
  } else {
    GPIO.write(pinLedB,1);
    isLedBOn = 1;
  }
}, null);
```

IoT application

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

IoT Devices Are Hacking Your Data & Stealing Your Privacy - Infographic

Ken Savage | 01.11.17 | iot, Infographic

Alexa beware! New smart home tests reveal serious privacy flaws

By Sandra Vogel - February 28, 2018

"Issues such as the fear of oversharing of data by commercial services, insufficient protection of stored personal data, and the possibility of interception of digital traffic by cybercriminals [are] significant."

When you live home ...

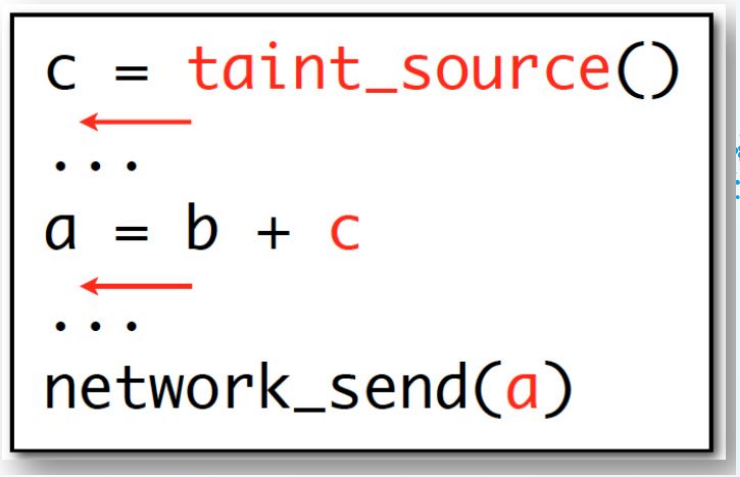
Whether the door is locked or not...

Saint: Tracking Sensitive Data in IoT Apps

- **Problem**: Users lack visibility into who sees their sensitive information
- Look inside of IoT apps to determine how they use privacy sensitive data
  - Device states
  - Device information
  - User inputs
  - Location

Saint: Tracking Sensitive Data in IoT Apps

- **Goal**: Analyze app source code to determine when privacy sensitive information leaves the IoT app
- Static taint analysis is a technique that tracks information dependencies from an origin
- Conceptual idea:
  - Taint source
  - Taint propagation
  - Taint sink

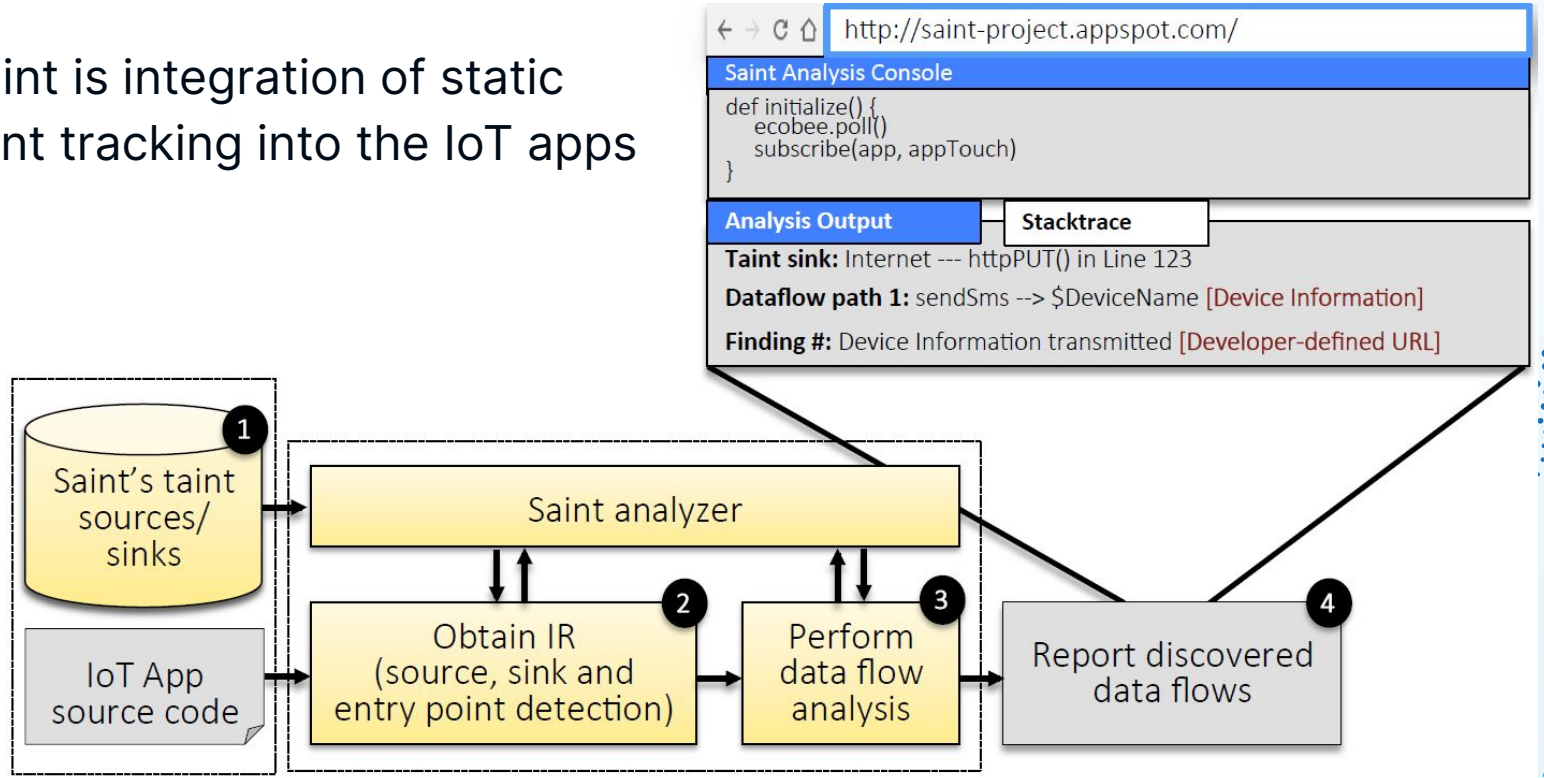


Challenges

- IoT programming platforms are diverse
- Identifying sensitive sources in IoT apps is quite subtle
- Each IoT platform is unique
  - has its idiosyncrasies that require special treatment

Saint

- Saint is integration of static taint tracking into the IoT apps



From app source code to IR

Devices

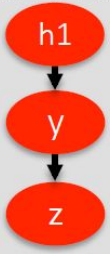
```
input (p, presenceSensor, type:device)
input (s, switch, type:device)
input (d, door, type:device)
input (toTime, time, type:user_defined)
input (fromTime, time, type:user_defined)
input (c, contact, type:user_defined)
```

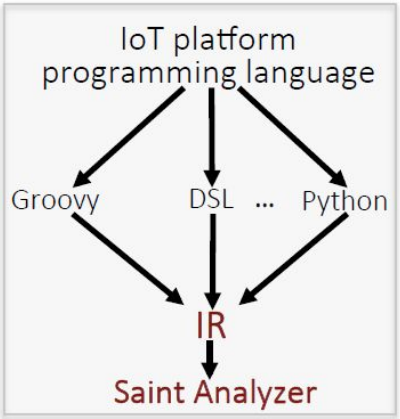
Events

```
subscribe(p, “present”, h1)
```

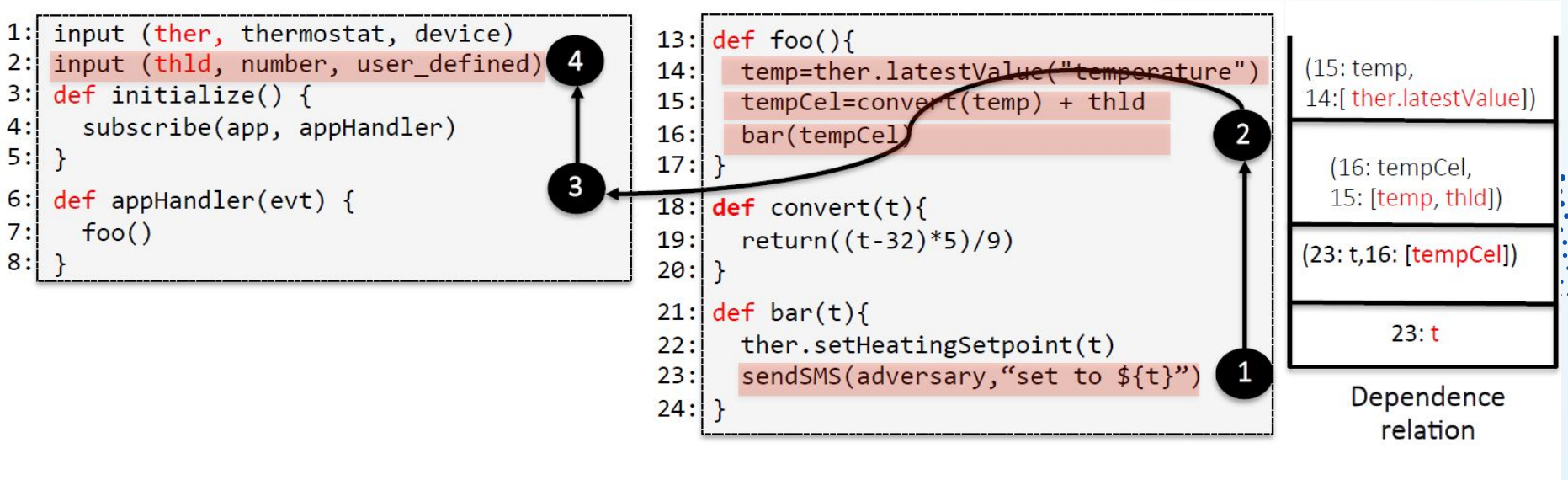
Computation

```
h1(){
        s.on()
        d.unlock()
        def between= y()
        if (between){
            z()
        }
}
y(){
        return timeOfDayIsBetween(fromTime, toTime)
}
z(){
        sendSms(c, “...”)
}
```





Backward taint tracking: Identify Sensitive Data Flow Paths



Analysis Sensitivity and Implicit Flows

- Path-sensitivity
  - Collects the evaluation results of the predicates
  - Discards infeasible paths
- Context-sensitivity
  - Implements depth-one call-site sensitivity
  - Discards paths not matching calls and returns
- Implicit flows
  - Determines whether predicates at conditional branches depends on a tainted value
  - Taints all elements in the conditional branch

Algorithms for IoT-specific idiosyncrasies

- **On-demand** algorithms for analysis precision

- State variables
  - Field-sensitive analysis
- Web service apps
  - Allows external entities to access devices
- Call by reflection
  - Add all methods as possible call targets

```
counter=state.switchCounter //state variable
if (counter){ device actions}
```

```
mappings { // web-service apps
  path("/switches") {
    action: [GET: "listSwitches"] }
def listSwitches() {
  return it.currentValue("switch")
}
```

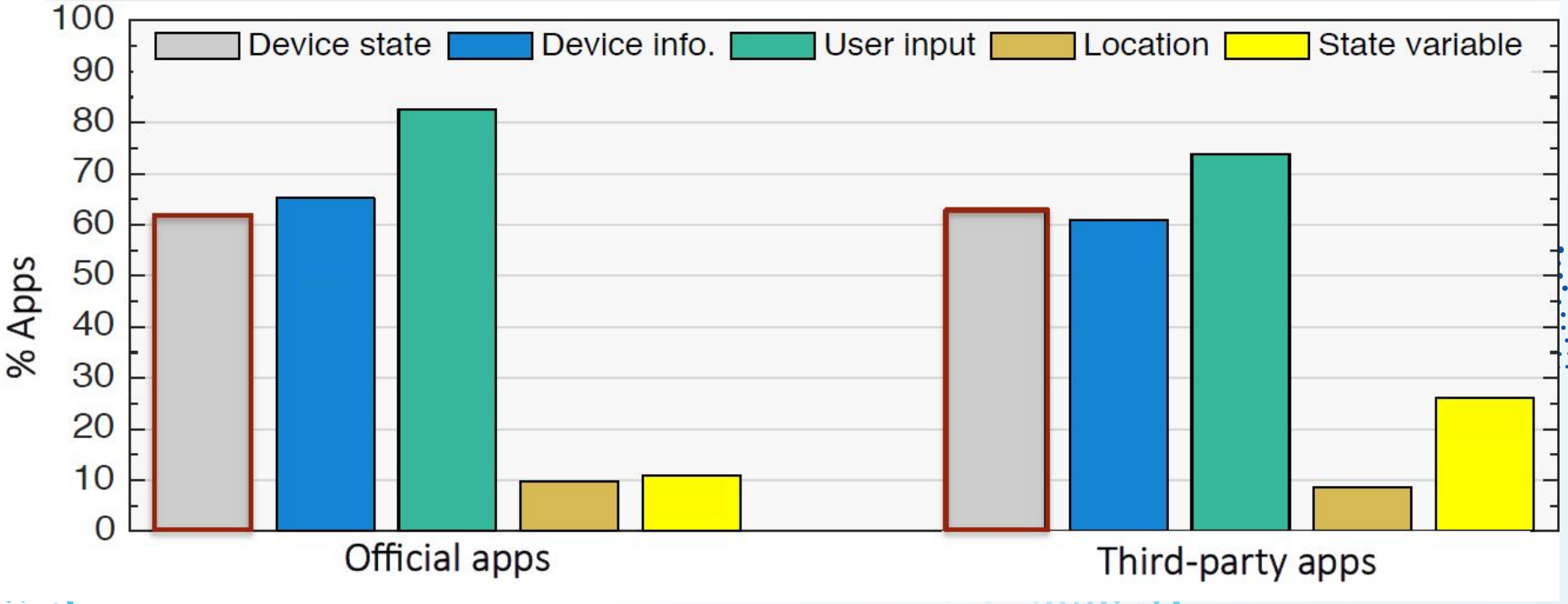
```
"$methodName"() // call by reflection

def foo() { // add as possible call target }
def bar() { // add as possible call target }
```

Application Study

- Implemented Saint for SmartThings IoT platform
- Selected **168** official and **62** third-party market apps
- **92** official and **46** third-party apps expose at least one kind of sensitive data

Apps	Internet	SMS	Both	Total
Official	24	63	5	92
Third-party	10	36	-	46

What type of privacy-sensitive information leaves IoT apps?



Who sees privacy-sensitive information?

Recipient Content

```
sendSMS(phoneNumber, “kids $presence ’’)
httpPost(URL, “kids $presence”)
```

		Recipient defined by			Content defined by		
Taint Sinks	Apps	User	Developer	External	User	Developer	External
Messaging	Official	154	0	0	5	149	0
	Third-party	67	0	0	4	63	0
Internet	Official	2	48	44	0	54	40
	Third-party	0	13	12	0	13	12

Thank you!

Questions?