Security Analysis of Emerging Smart Home Applications

Earlence Fernandes, Jaeyeon Jung, Atul Prakash

IEEE Security and Privacy
24 May 2016

CO Sensors

Smart Door Locks

Connected Ovens

Smart Plugs

IP Cameras

Smart TVs

APPS

WEAVE On WEAVE

Google Brillo inside

SAMSUNG SmartThings

HomeKit

Emerging Smart Home Frameworks

Potential Security Risks

Remotely determine prime time for Burglary [1,2]

OR

Flooding [1]

[1] Denning et al., Computer Security and the Modern Home, CACM'13

[2] FTC Internet of Things Report'15

Current Vulnerabilities

Devices

Protocols

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

In what ways are these emerging, *programmable* smart homes vulnerable to attacks, and what do those attacks entail?

Analysis of SmartThings

Access Control

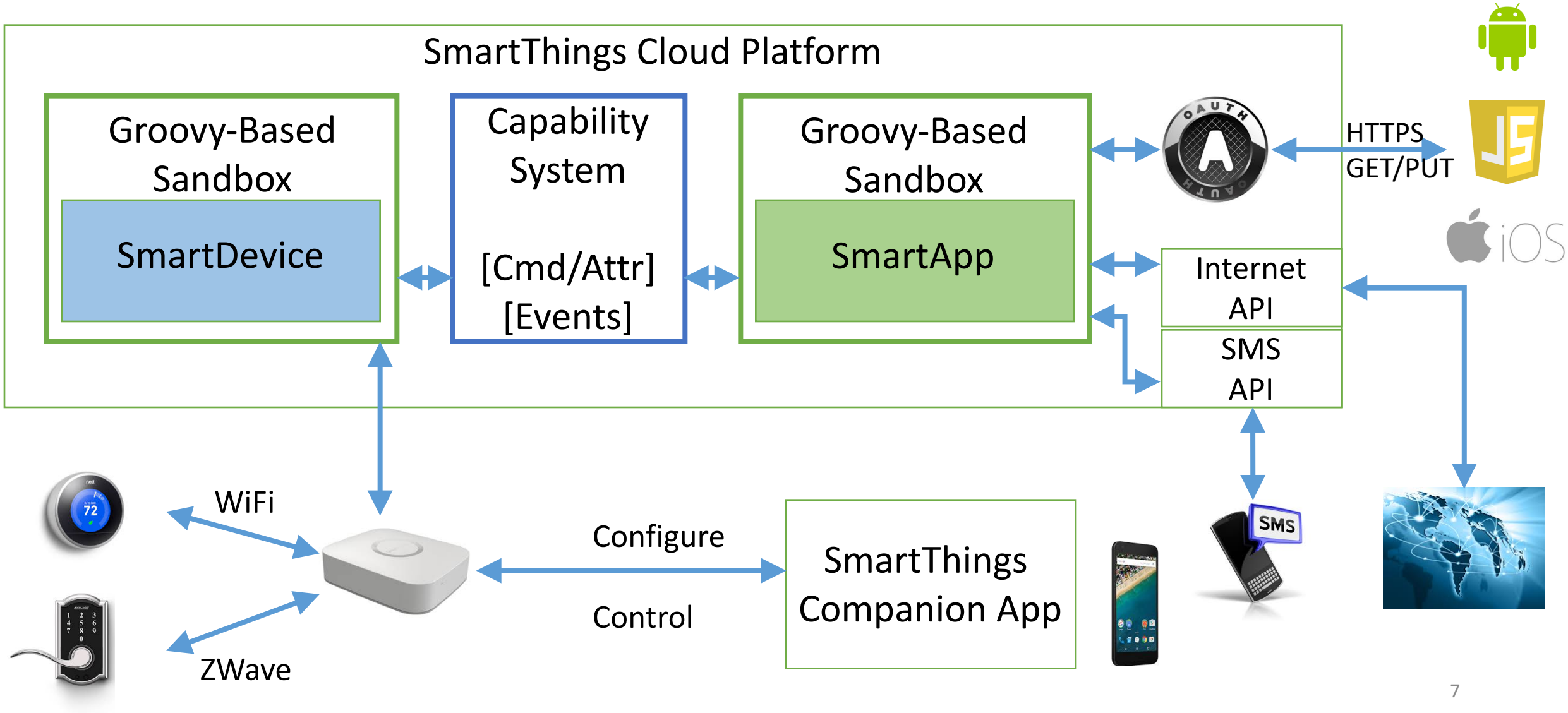
Trigger-Action Programming

- **Why SmartThings?**
  - Relatively Mature (2012)
  - 521 SmartApps
  - 132 device types
  - Shares 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

Analysis of SmartThings – 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

Untrusted SmartApp

Send commands

Read/set attributes

Receive events

ZWave Lock SmartDevice

capability.lock
capability.lockCodes
capability.battery
…

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

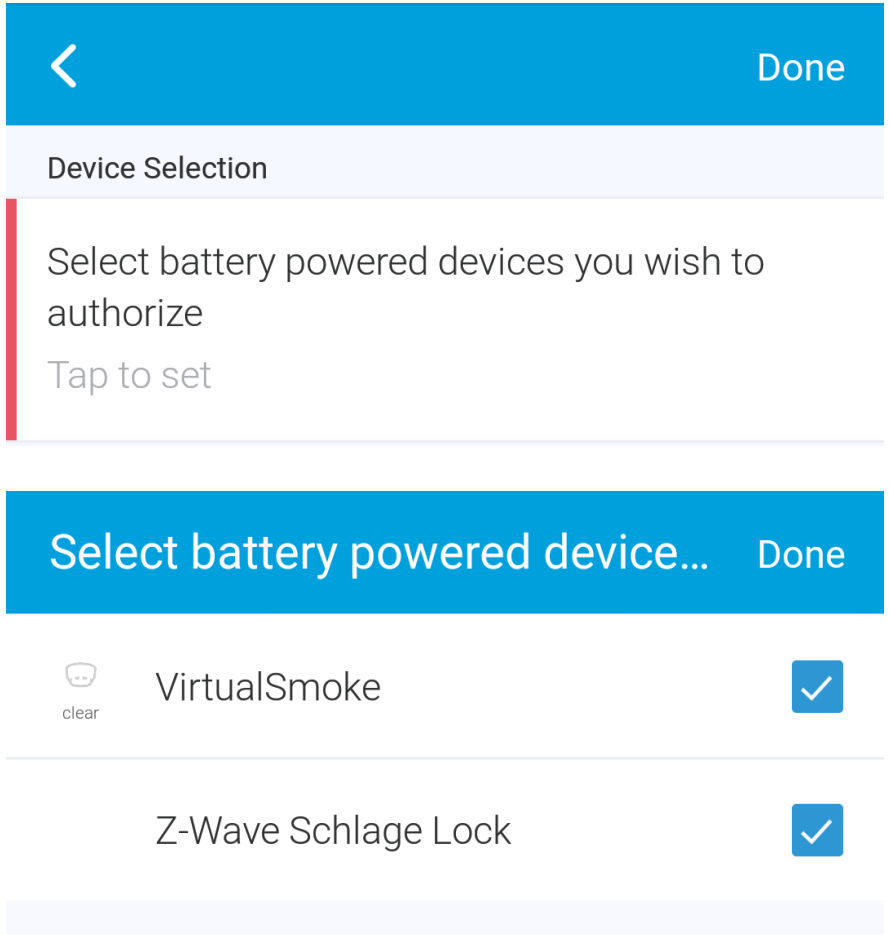
Usability
Simpler Coarser Capabilities

Ease of Development
Expressive Functionality

Security
Very Granular Capabilities

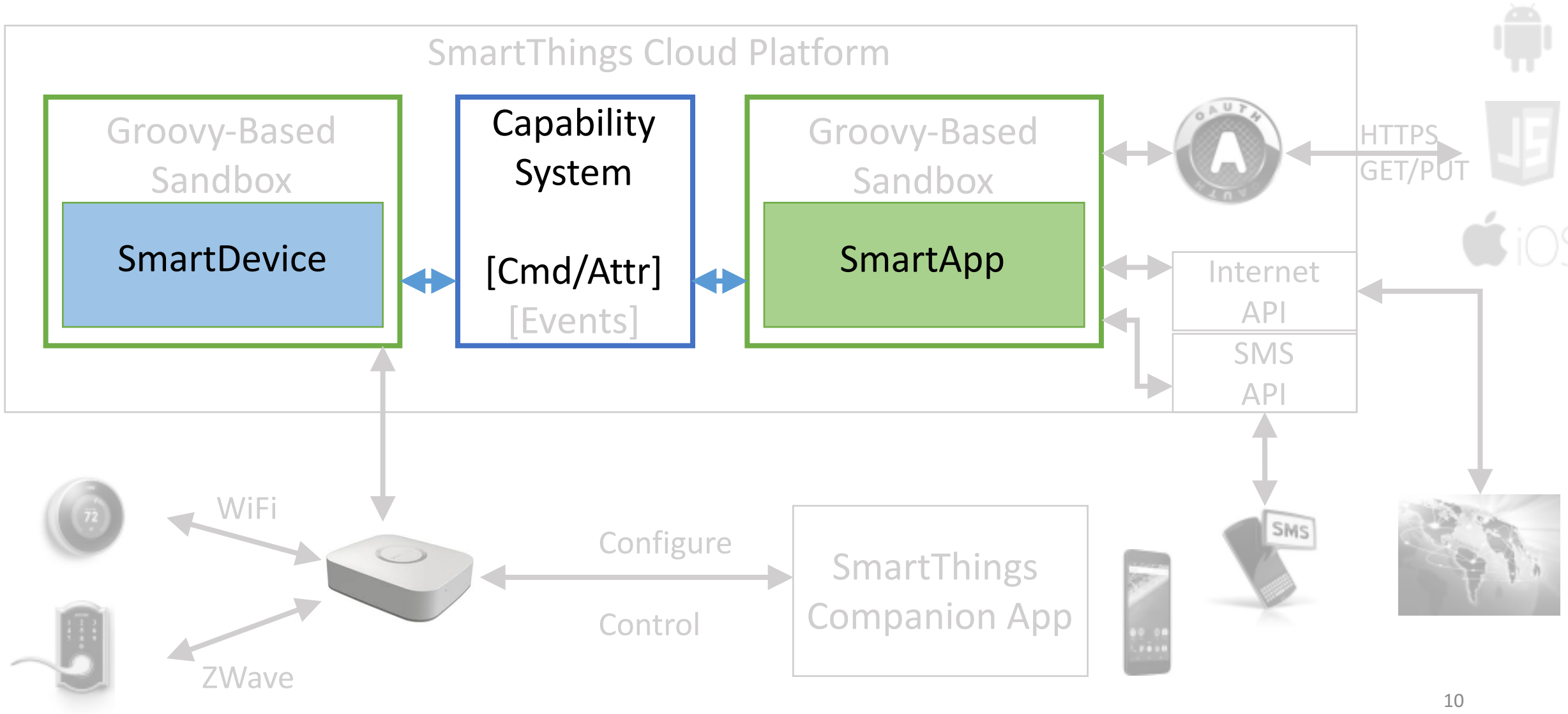
SmartApps request Capabilities

```
definition(name: “DemoApp”,
namespace: “com.testing”, category: “Utility”)

//query the user for capabilities
preferences {
   section(“Battery-Powered Devices”) {
      input “dev”, “capability.battery”, title: “Select
      battery powered devices you wish to authorize”,
      multiple: true
   }
}

…
```



Device Enumeration

Overprivilege in SmartApps



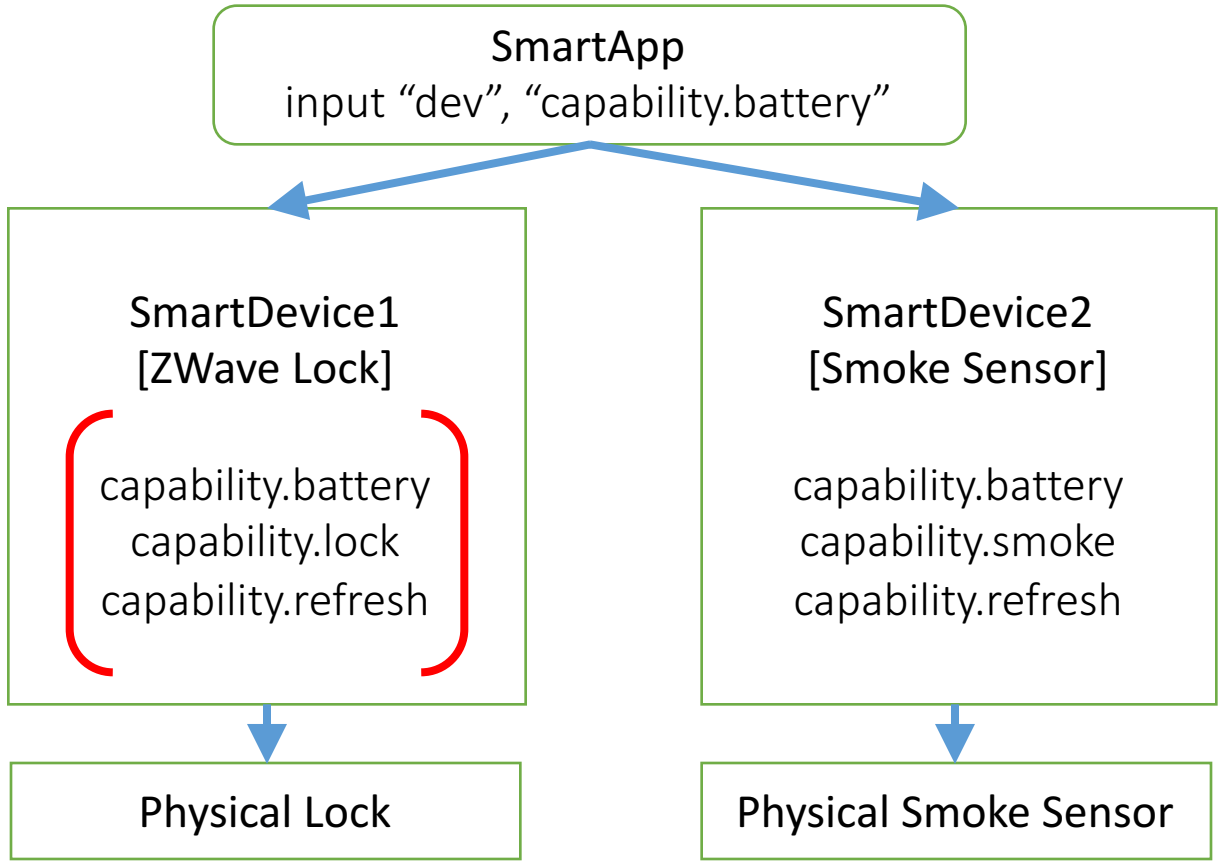
Overprivilege in SmartApps

Coarse-Grained Capabilities

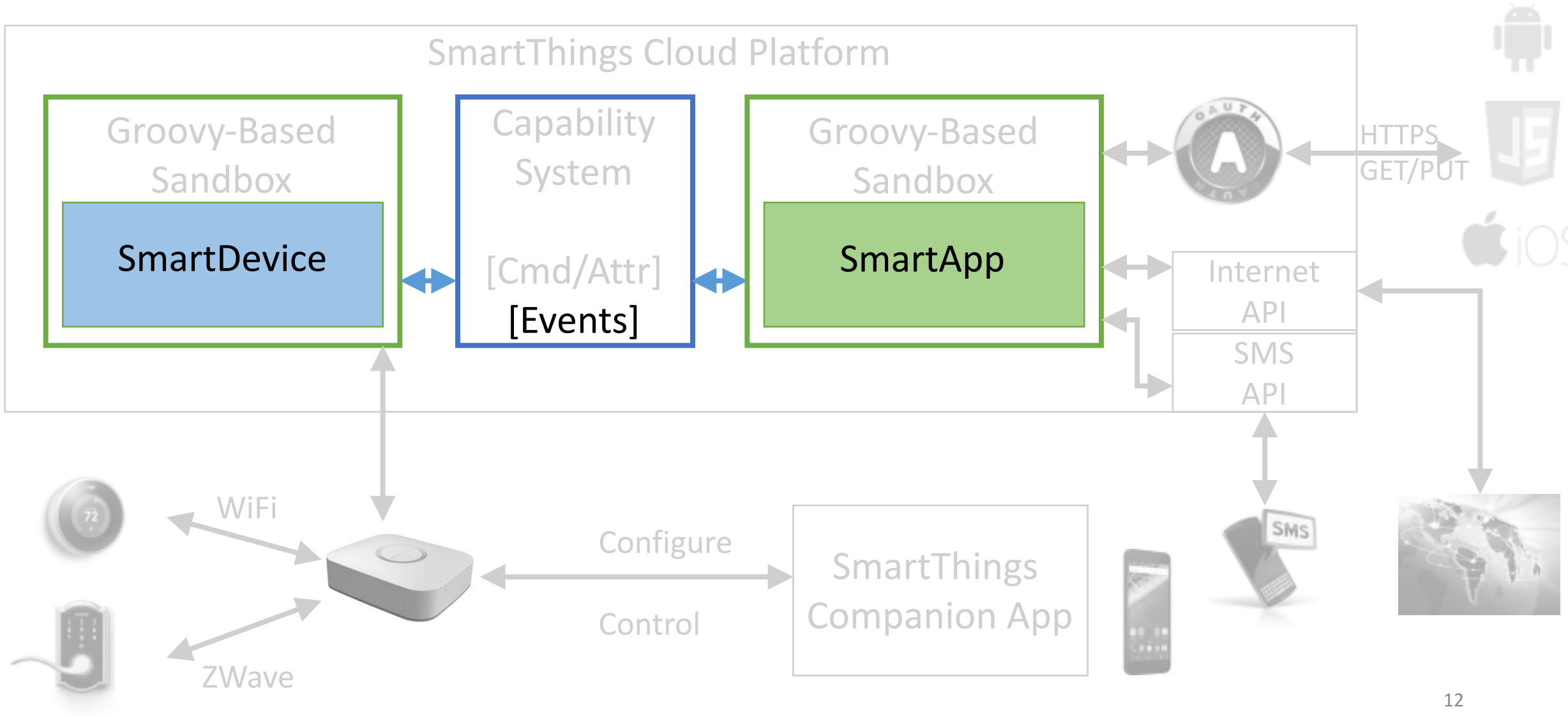
- “Auto-lock” app from app store
- Only needs “lock” command, but can also issue “unlock”

Overprivilege Increases Attack Surface of the Home

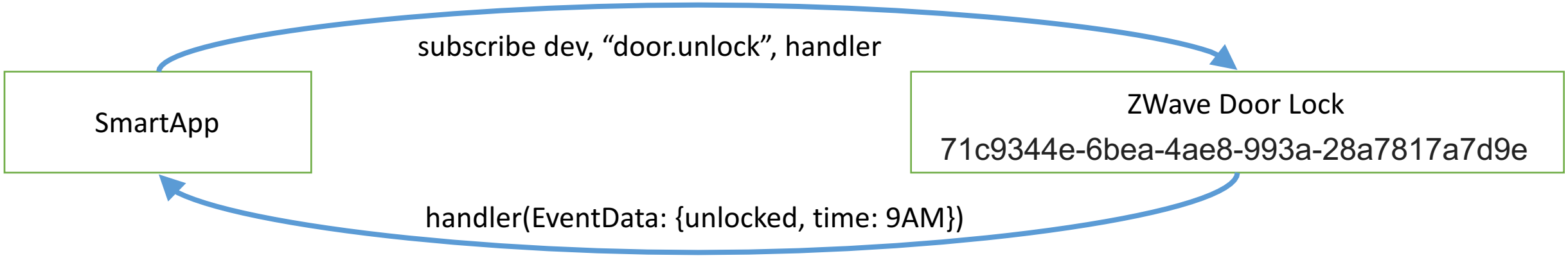
Coarse SmartApp-SmartDevice Binding



Insufficient Event Data Protection

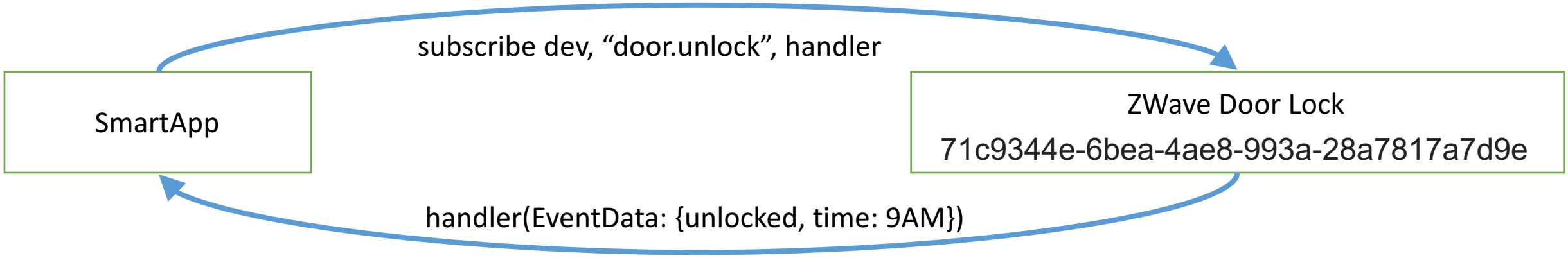


Insufficient Event Data Protection



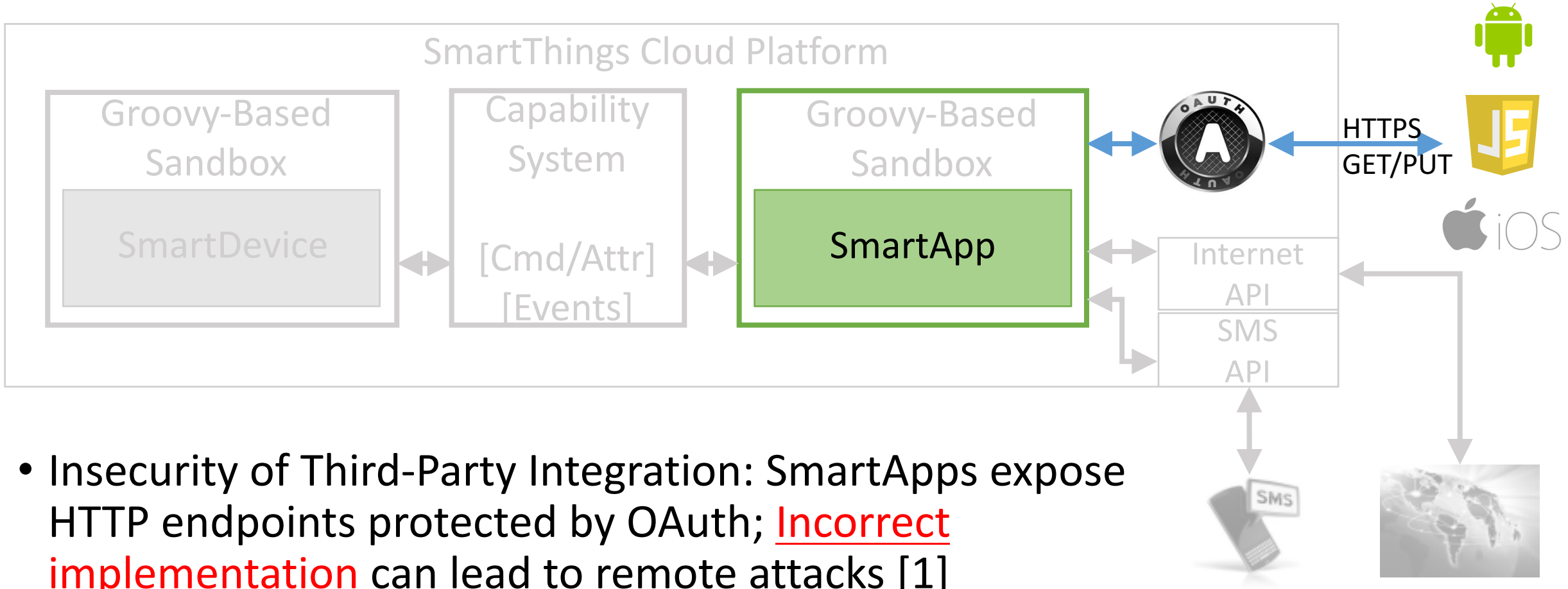
- Once a SmartApp gains any capability for a device, it can subscribe to any event that device generates
- If a SmartApp acquires the 128-bit ID, then it can monitor all events of that device without gaining any of the capabilities the device supports
- Using the 128-bit ID, a SmartApp can spoof physical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

Insufficient Event Data Protection



- Can lead to leakage of confidential information
- Spoofed Events can lead to Apps/Devices taking incorrect actions

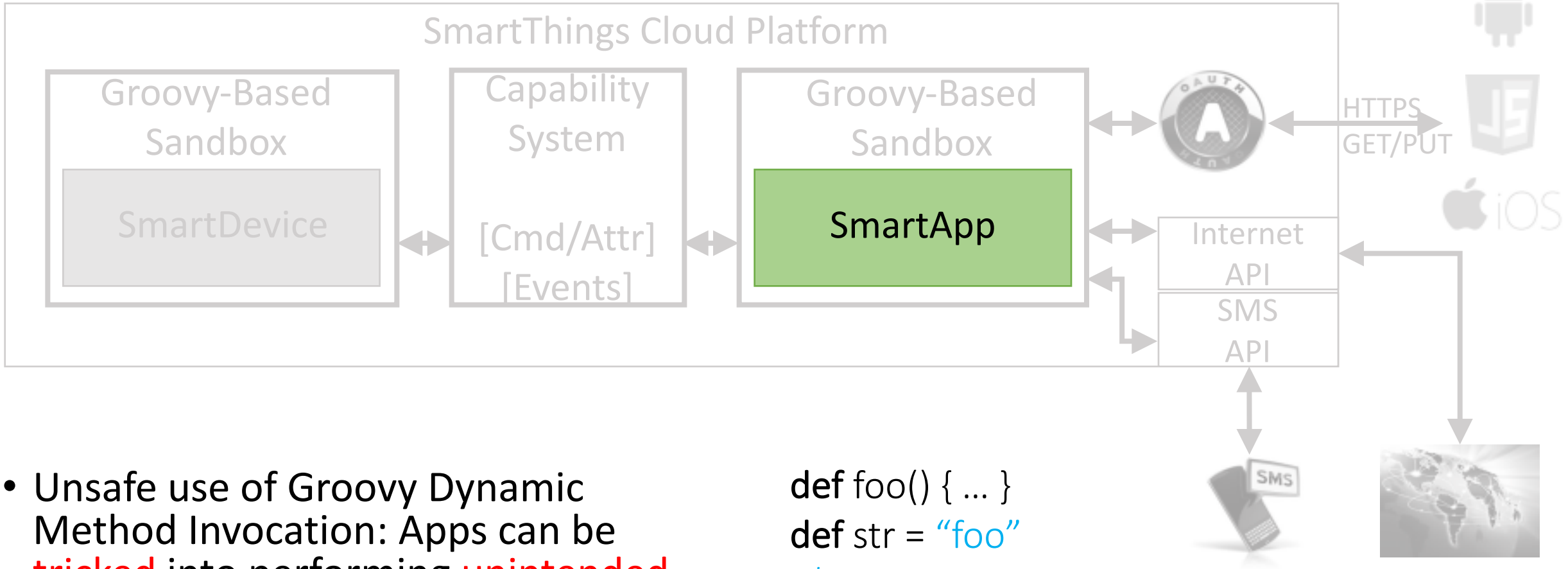
Other Potential Security Issues - OAuth



- 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

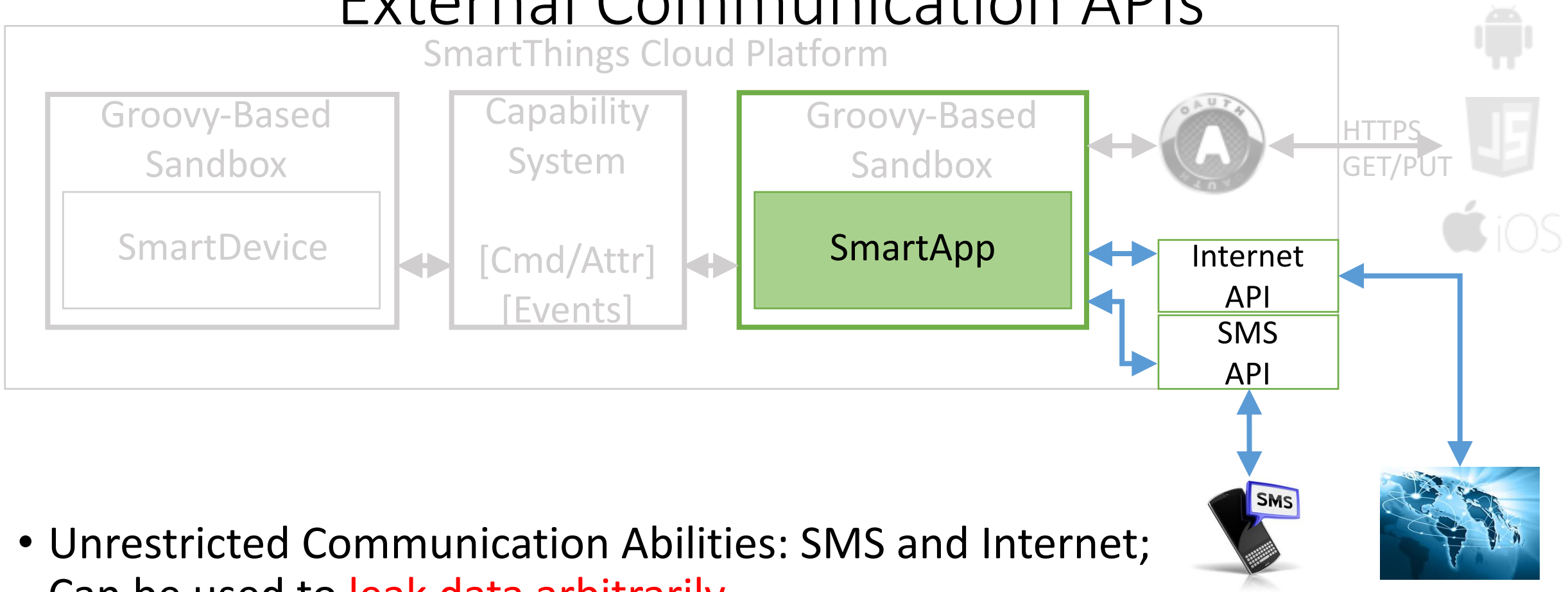
Other Potential Security Issues



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

```
def foo() { … }
def str = “foo”
“$str”()
```

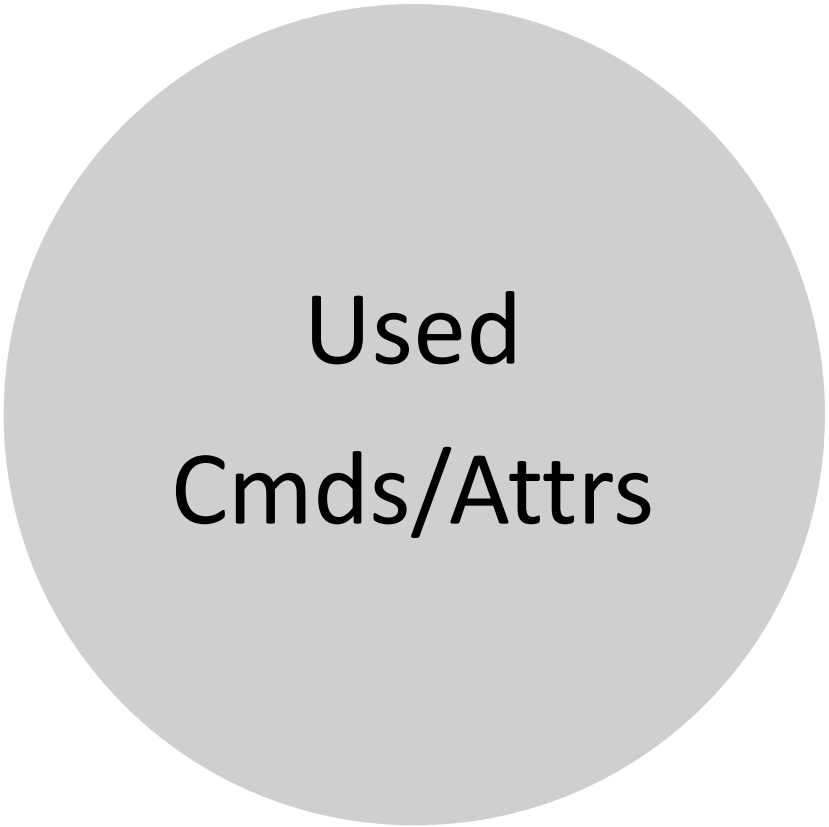
Other Potential Security Issues – Unrestricted External Communication APIs



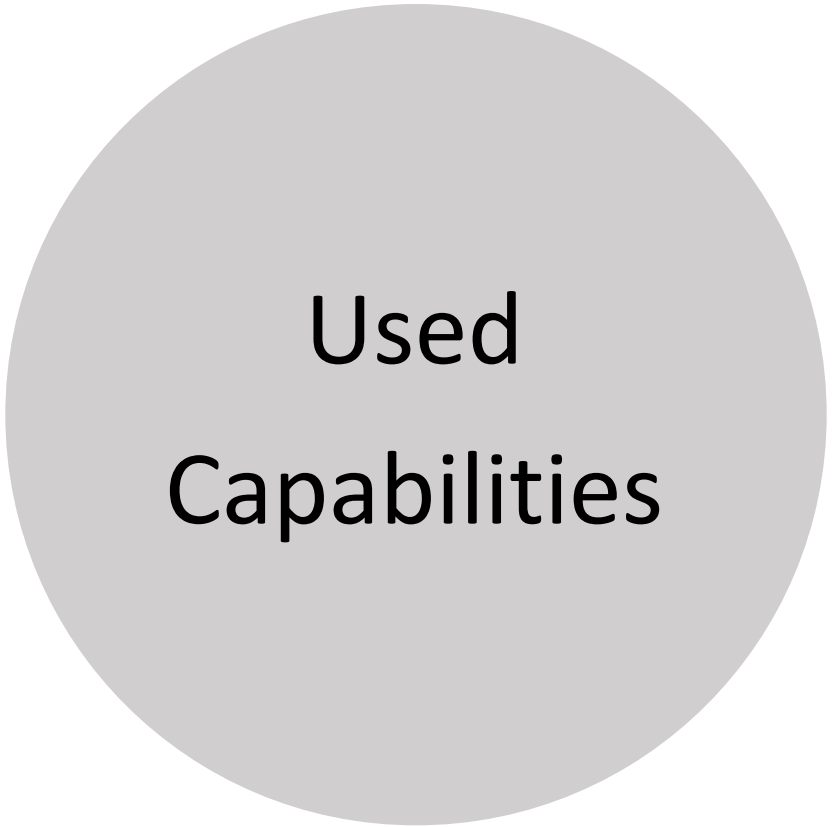
- Unrestricted Communication Abilities: SMS and Internet; Can be used to leak data arbitrarily

Computing Overprivilege

Coarse-Grained Capabilities



Coarse SmartApp-SmartDevice Binding



Measuring Overprivilege in SmartApps

Challenge

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

Solution

- 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

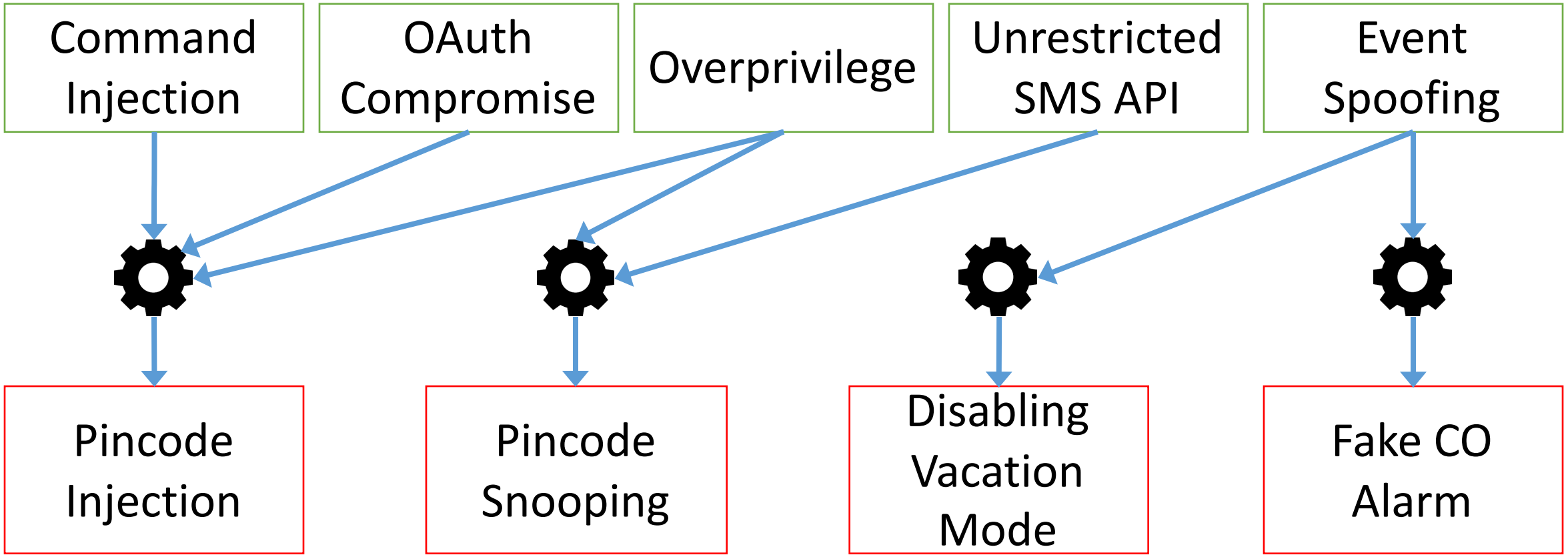
Empirical Analysis Results

	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

[1] Felt et al., I've got 99 problems, but vibration ain't one: A survey of smartphone users' concerns, SPSM'12

Summary

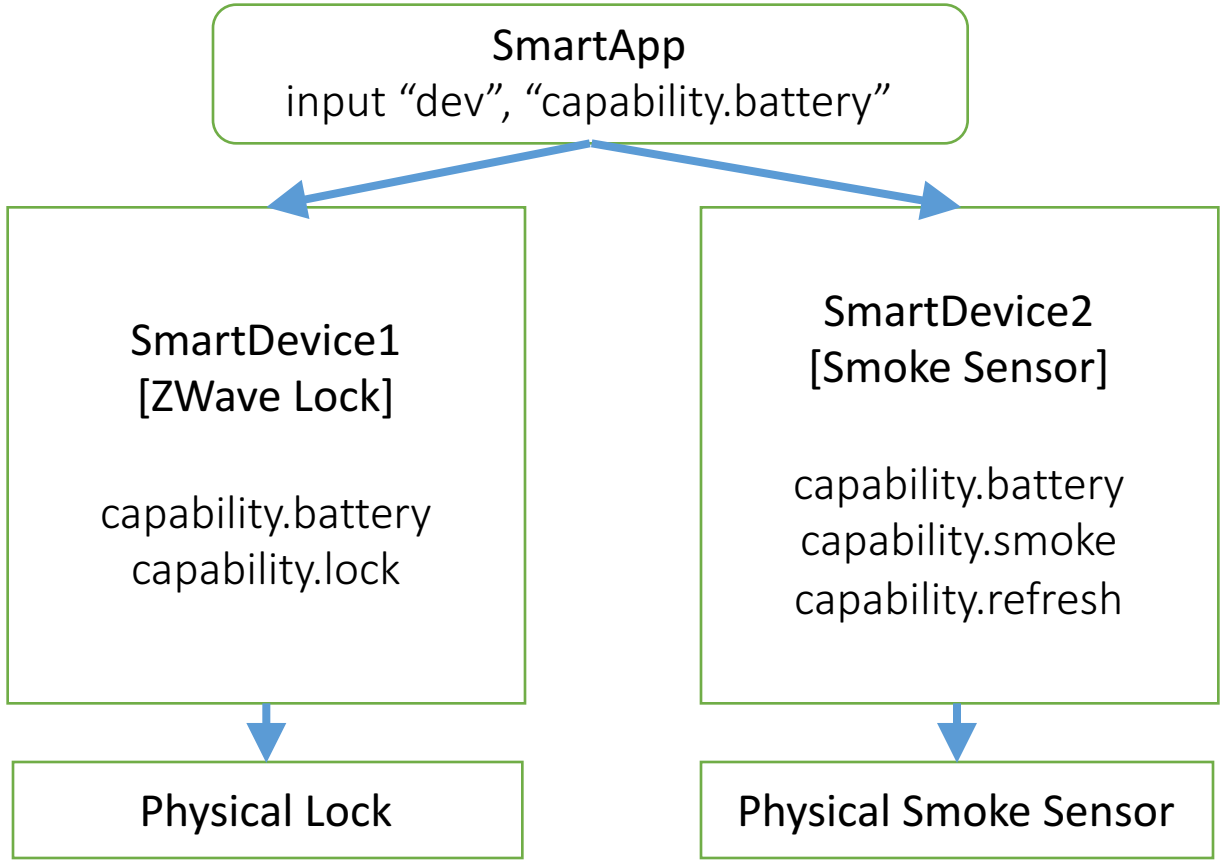
- **First look at the security design of a programmable smart home platform**: Samsung SmartThings; Challenge: Blackbox Cloud System
- **Two security design issues**:
  - Overprivilege: Coarse grained capabilities, and Coarse SmartApp-SmartDevice Binding
  - Insecure Events: Apps do not need special privileges to access sensitive info
- **Empirical Analysis**: 55% of apps do not use all operations their capabilities imply; 43% get capabilities they did not explicitly request
- **Four PoC attacks that combine various security design issues**
  - These attacks are device independent, and long-range
- **Security Improvements:** Notified SmartThings in Dec 2015; Improvements in vetting process and developer best practices for Groovy Strings (Apr 2016); Discussion on improvements to capability system (May 2016)

Security Analysis of Emerging Smart Home Applications

- **First look at the security design of a programmable smart home platform**: Samsung SmartThings; Challenge: Blackbox Cloud System
- **Two security design issues**:
  - Overprivilege: Coarse grained capabilities, and Coarse SmartApp-SmartDevice Binding
  - Insecure Events: Apps do not need special privileges to access sensitive info
- **Empirical Analysis**: 55% of apps do not use all operations their capabilities imply; 43% get capabilities they did not explicitly request
- **Four PoC attacks that combine various security design issues**
  - These attacks are device independent, and long-range
- **Security Improvements:** Notified SmartThings in Dec 2015; Improvements in vetting process and developer best practices for Groovy Strings (Apr 2016); Discussion on improvements to capability system (May 2016)

Conservatively Statically Estimating SmartApp-SmartDevice Overprivilege

- Many devices can implement a given capability
- Statically, we do not which device the user would assign to an app
- Use our dataset of 132 device handlers to estimate, conservatively



Empirical Analysis of SmartThings

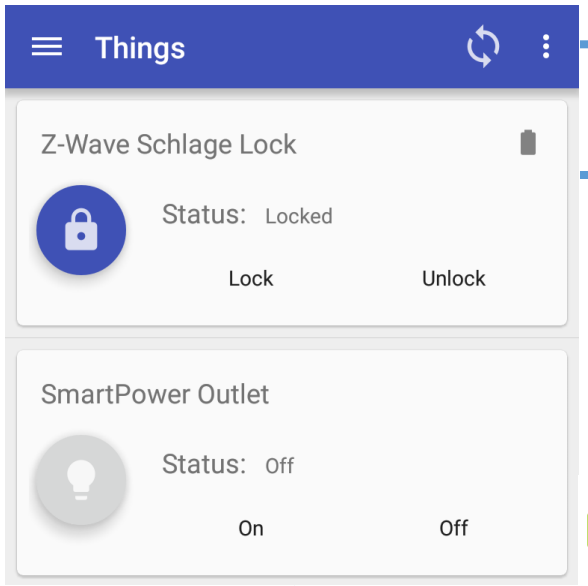
Total number of SmartDevices	132
Number of SmartDevices raising events using createEvent and sendEvent. Such events can be snooped on by SmartApps	111

Total number of SmartApps	499
Number of apps using potentially unsafe Groovy dynamic method invocation	26
Number of OAuth-enabled apps, whose security depends on correct implementation of OAuth	27
Number of apps using unrestricted SMS APIs	131
Number of apps using unrestricted Internet APIs	36

Exploiting Design Flaws in SmartThings

Attack Description	Attack Vectors	Physical World Impact
Backdoor Pincode Injection Attack	Command injection into existing WebService SmartApp; Overprivilege; OAuth impl. flaws	Enabling physical entry; Theft
Door Lock Pincode Snooping Attack	Stealthy battery-level monitoring app; Overprivilege; leak data using SMS	Enabling physical entry; Theft
Disabling Vacation Mode Attack	Attack app with no capabilities; Misusing logic of benign app; Event Spoofing	Theft; Vandalism
Fake Alarm Attack	Attack app with no capabilities; Event spoofing; Misusing logic of benign app	Misinformation; Annoyance

Backdoor Pincode Injection Attack



HTTP PUT

HTTP GET

WebService SmartApp

client_id
client_secret

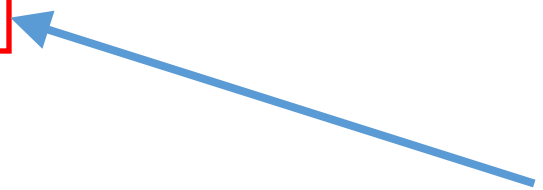
```
mappings {
   path("/devices/:id") { action:
   [ PUT: "updateDevice" ]
}

def updateDevice()
{
   def cmd = request.JSON.command
   def args = request.JSON.arguments
   // code truncated
   device."$cmd"(*args)
}
```

```
{
   command: setCode,
   arguments: [3, '5500']
}
```

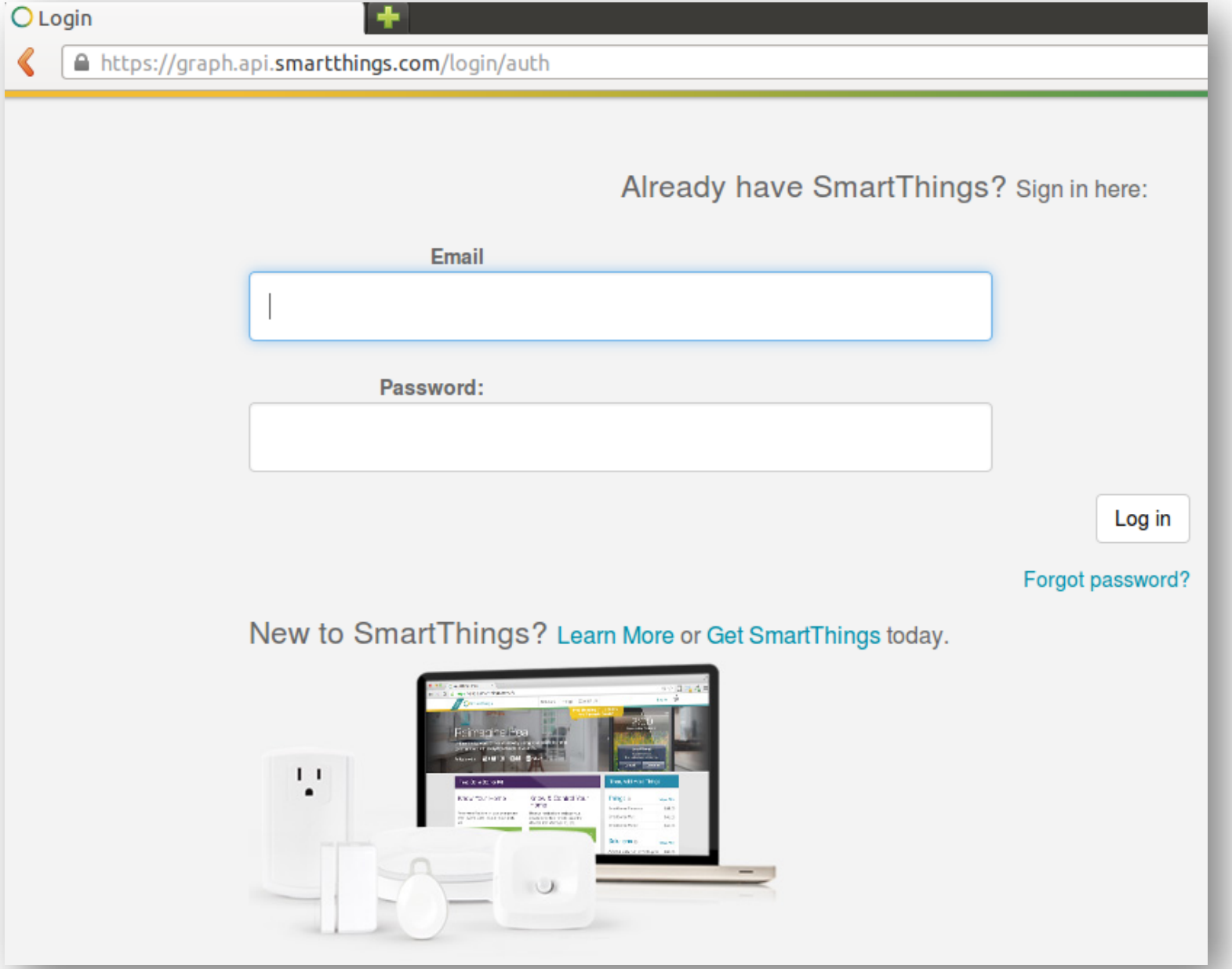
Example of Stealing an OAuth Bearer Token

- Decompile APK bytecode to get the client_secret
- Send email to user asking to “reauthenticate” to SmartThings

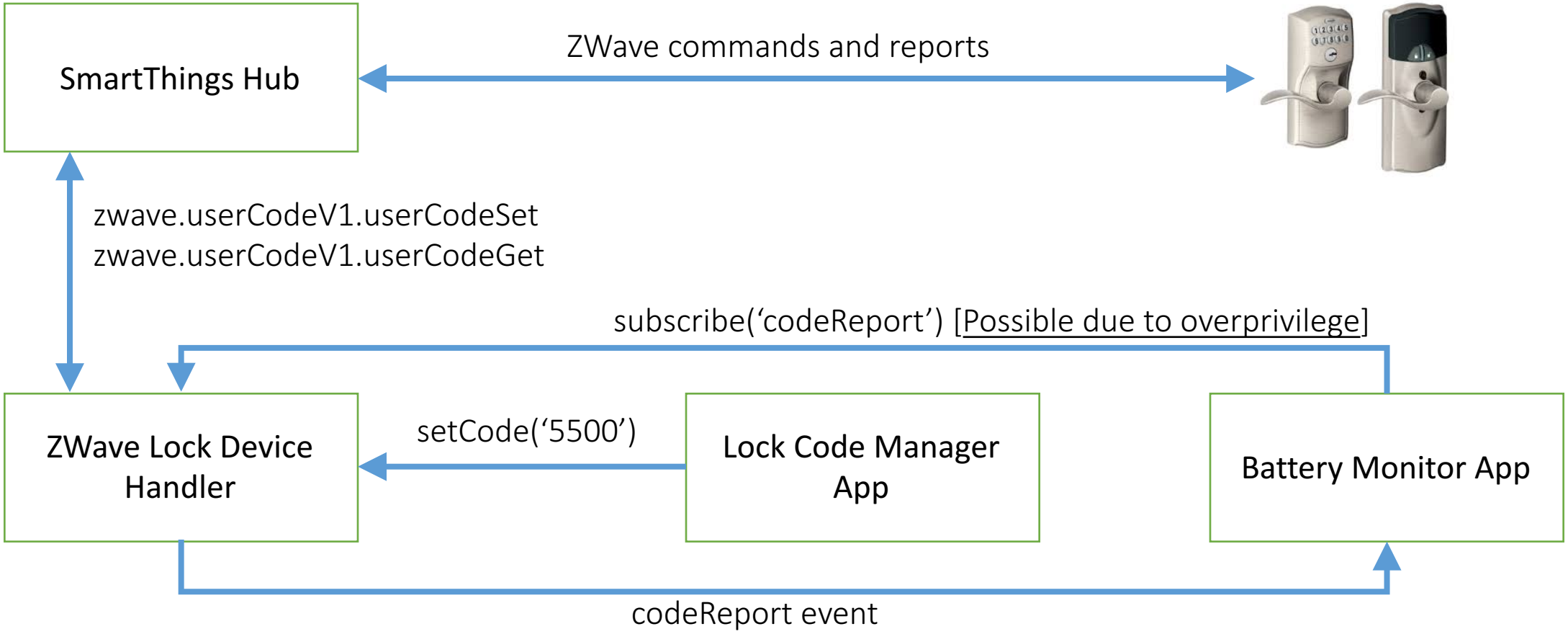
https://graph.api.smartthings.com/oauth/authorize?response type=code&client_id=REDACTED&scope=app&
redirect_uri=http%3A%2F%2Fssmartthings.appspot.com



Open Redirector


Login
https://graph.api.smartthings.com/login/auth
Already have SmartThings? Sign in here:
Email
Password:
Log in
Forgot password?
New to SmartThings? Learn More or Get SmartThings today.

Door Lock Pincode Snooping Attack



Responsible Disclosure

Dec 17, 2015
We contacted SmartThings with details on attacks.

Jan 12, 2016
SmartThings acknowledged the attacks and said they are working on solutions.

Apr 15, 2016
SmartThings informed us that docs were updated to recommend filtering Groovy Strings; Vetting processes were updated to look for our attacks.

May 2, 2016
We had a call with SmartThings team to discuss potential new design for capability system.

Emerging Smart Home Frameworks

Google

WEAVE On WEAVE

Brillo inside

SAMSUNG

SmartThings

vera™

Smarter Home Control

HomeKit

AllJoyn™

amazon alexa

Current Vulnerabilities in Smart Homes

Devices

Protocols

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

IP Cameras

APPS

Smart Door Locks

Smart Plugs

CO Sensors

Smart TVs

Connected Ovens