ContexIoT: Towards Providing Contextual Integrity To Appified IoT Platform

Yunhan Jack Jia, Qi Alfred Chen, Shiqi Wang, Amir Rahmati,
Earlence Fernandes, Z.Morley Mao, Atul Prakash

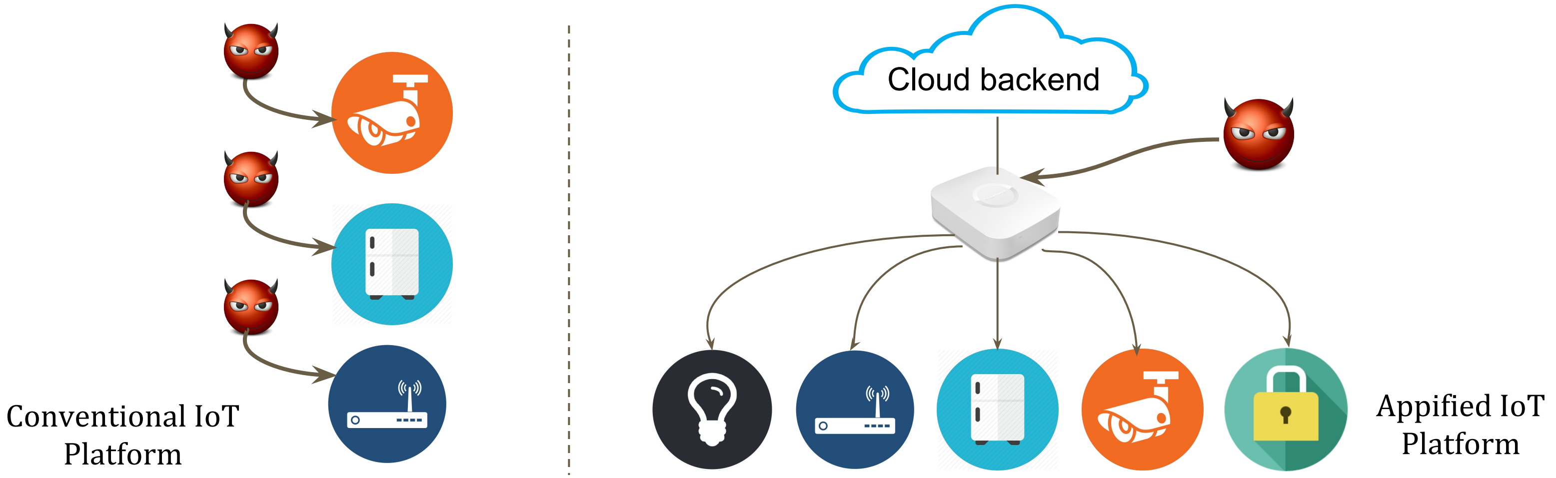
University of Michigan

Appified Platform

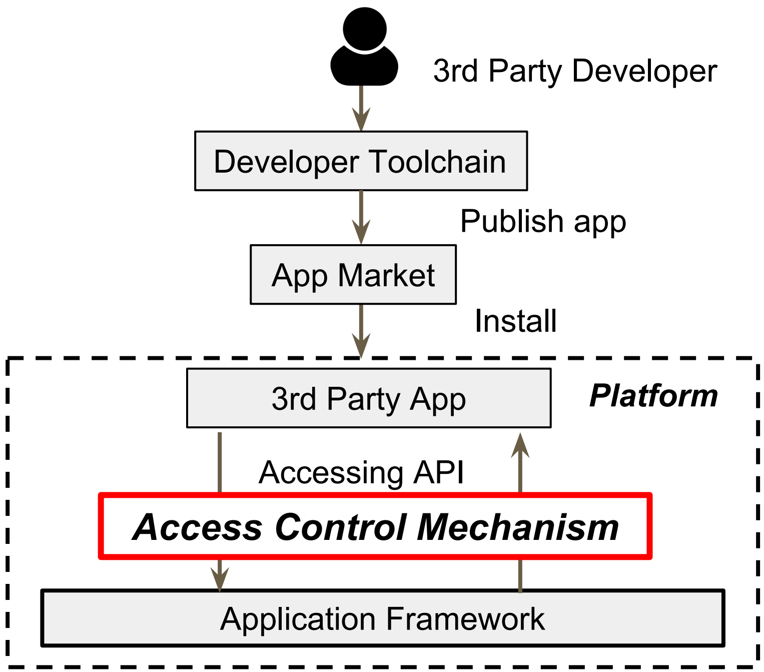
- Software platform that supports 3rd-party app development

IoT

- Emerging IoT threats (*conventional* **vs.** *appified*)



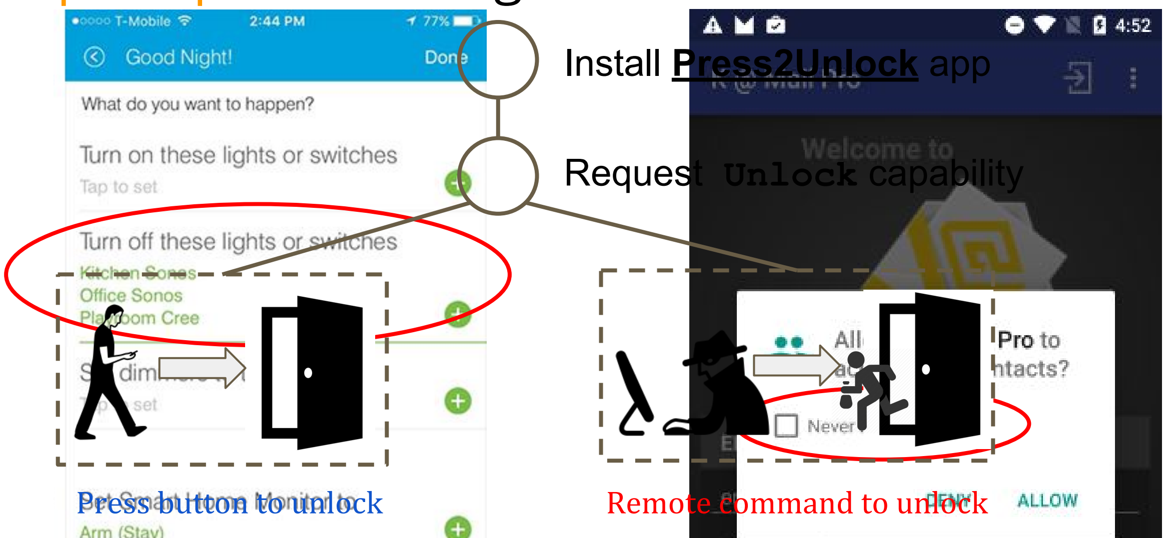
The Ecosystem and Threats



- **Problem** : allowing untrusted app to control the user's device
- **Solution** : access control system (***Permission system***)

Permission System: Practices & Limitations

- Installation time prompts: decision made out-of-context
- Runtime prompts: coarse-grained and insufficient



IoT malware example

Ideal access control needs to be in context and sufficiently fine-grained

Contextual Integrity

- First appeared in "Washington Law Review 2004"
  - Advocated as a benchmark for privacy
- Contextual integrity for access control
  - *A permission granted to an app shall only allow the app behavior under that specific context, in which the permission is granted.*

$$Context(T_{using}) == Context(T_{granted})$$

- Requirements for the context:
  - **Distinguishable** : differentiate context upon permission request
  - **Meaningful** : tell benign from malicious

1. **How to define context in an access control system for IoT?**
2. **How to enforce its integrity in the IoT apps?**

Attack Taxonomy Methodology

Reported IoT attacks

Pin code snooping

Get remote shell

……

BLE relay unlocking

12

Mobile adversary techniques

Repackaging

Shadow payload

……

Side channel

17

Evaluate attacks against context definitions in related work

Context components

Control flow

Data flow

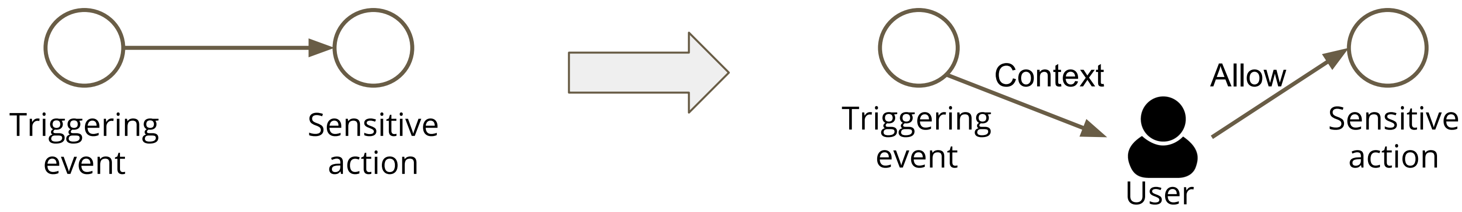
……

Context Definition

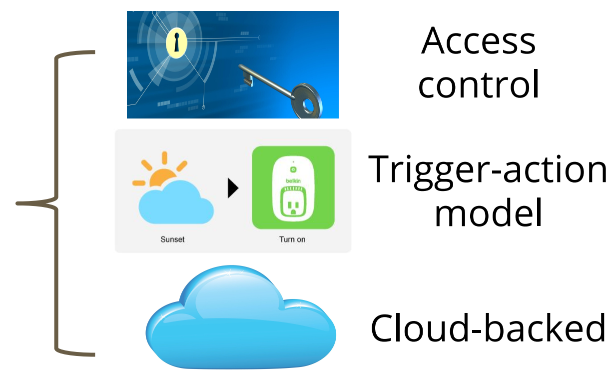
Related work	Context components					Decision made in context?
	UID/GID	UI Activity	Control flow	Runtime value	Data flow	
ACG	✔	✔	✘	✘	✘	✔
CRePE	✔	✘	✘	✔	✘	✘
AppContext	✔	✘	✔	✘	✔	✘
AppFence	✔	✘	✘	✘	✔	✘
Aurasium	✔	✔	✘	✔	✘	✔
FlaskDroid	✔	✘	✘	✔	✘	✘
SEAndroid	✔	✘	✘	✘	✘	✘
SEACAT	✔	✔	✘	✔	✘	✔
TaintDroid	✔	✔	✘	✔	✔	✔
TriggerScope	✔	✘	✔	✘	✔	✘
ContexIoT	✔	N/A	✔	✔	✔	✔

ContexIoT Design Goal

- Objective: *Prompt user with essential context to grant access to desired app behavior*

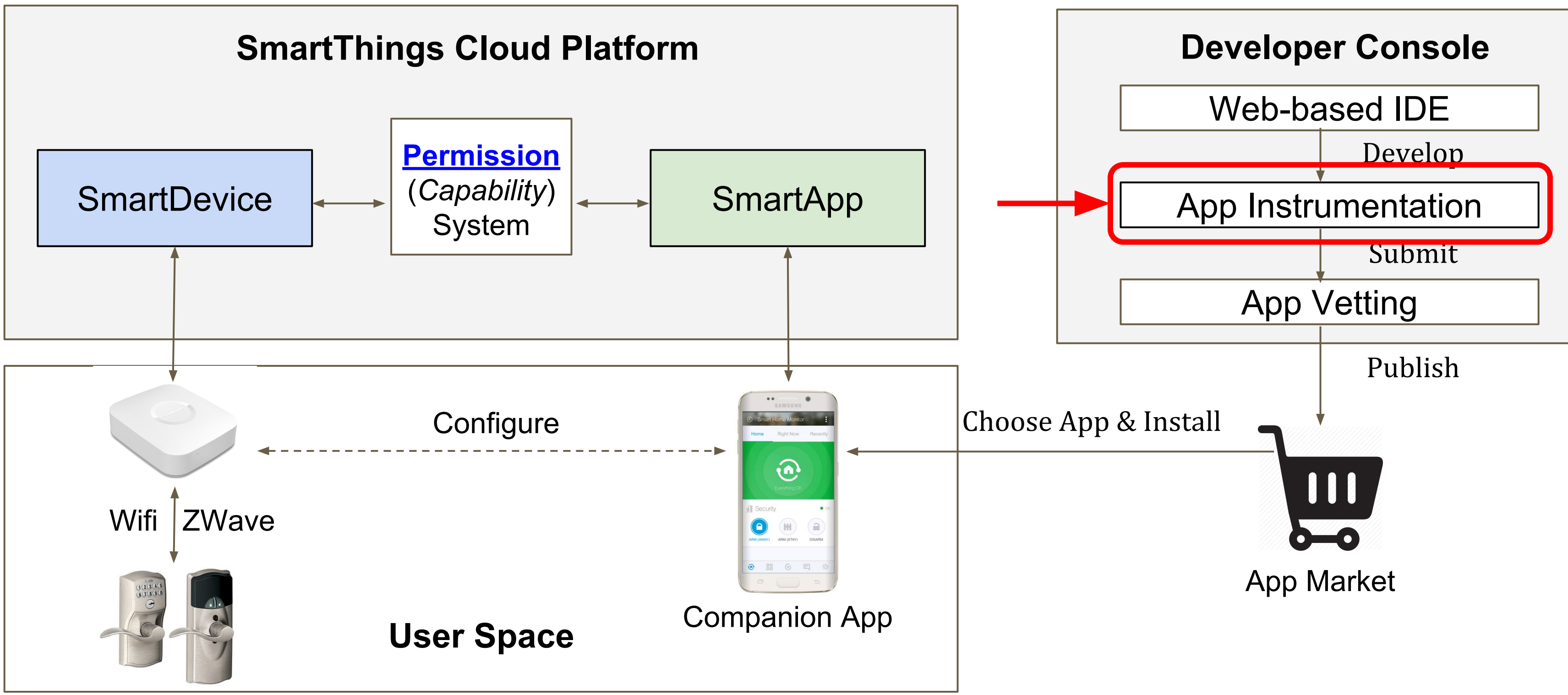


- Why *SmartThings*:
  - Relatively mature: 500~ apps, 150~ devices
  - Sharing design principles with other platforms



- Design challenge: limited access to the cloud-backed IoT platform

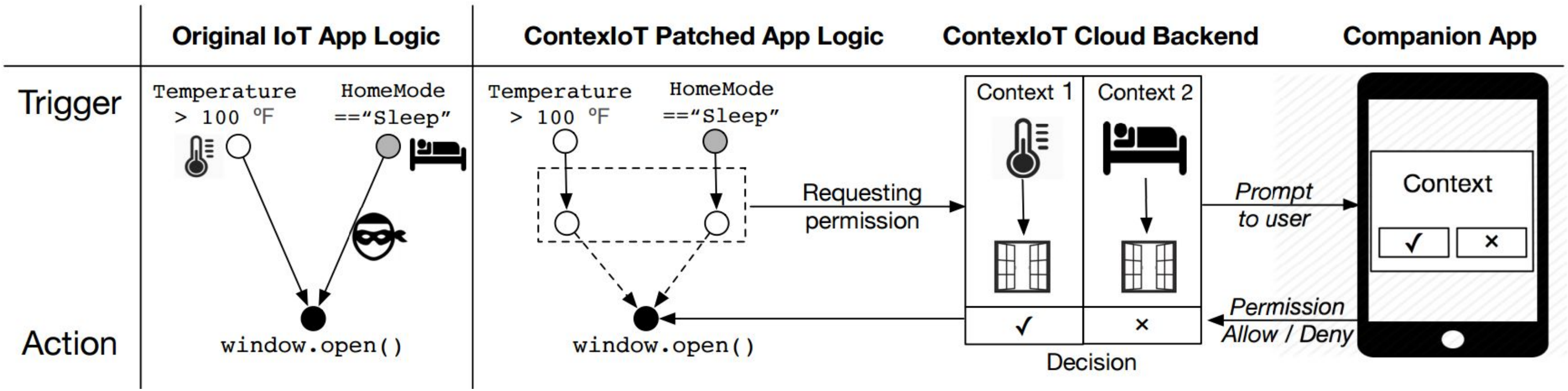
Design: SmartThings Background



Cloud-backed SmartThings Ecosystem

ContexIoT Design

- Patch SmartApps with the access control logic



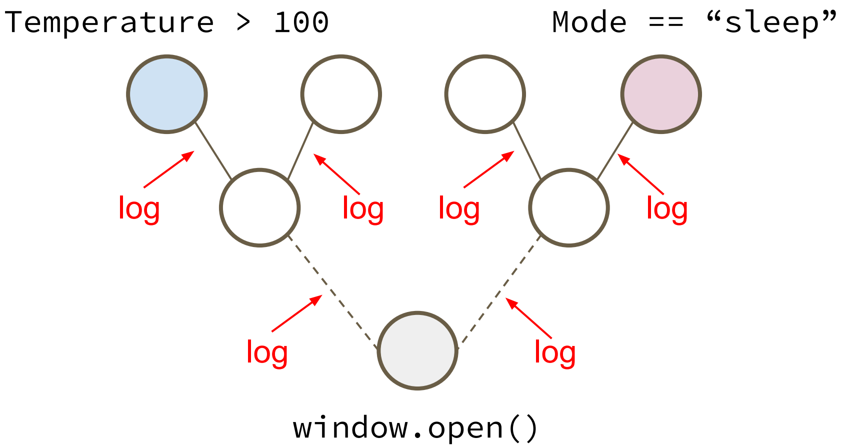
$$\boxed{Context(T_{using})} == Context(T_{granted})?\ \boxed{\texttt{Reuse() : Prompt()}}$$

Context collection logic

End-to-end secure logic

Implementation: Context Collection

- Patch SmartApp with context collection logic
  - Naive approach: add runtime logging to the entire instruction set
  - To reduce runtime overhead, use offline static analysis to:
    - **Prune**: Identify a subset of app code that requires runtime logging
    - **Precompute**: calculate context information that is deterministic



Static Analysis: Precompute Context

- Precompute intra-procedure context that are deterministic
  - Chained later to construct complete context according to the call trace
  - **GString** -- dynamic feature in Groovy

```
def temperatureHandler(evt)
{
    if(100<evt.value){
        window.open() //sink1
    }
}
def requestHandler(evt)
{
    def command = evt.params
    if("open" == command){
        window.open() //sink2
    }
}
```

```
def intra_context_sink1="[
    {type: 'func', name: 'temeratureHandler', param: evt},
    {type: 'ifstmt',
        condition:[{left: evt.value, right: 100, op: '<'}],
        branch: true},
    {type: 'sink', entity: window, action: open, param: null}
]"
```

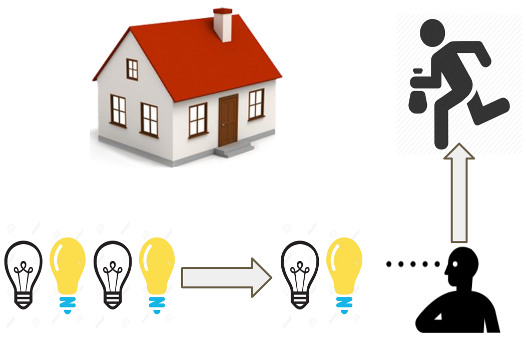
Open window when temperature goes high above 100

```
def intra_context_sink2="[
    {type: 'func', name: 'requestHandler', param: evt},
    {type: 'ifstmt',
        condition:[{left: command, right: 'open', op: '=='}],
        branch: true},
    {type: 'sink', entity: window, action: open, param: null},
]"
```

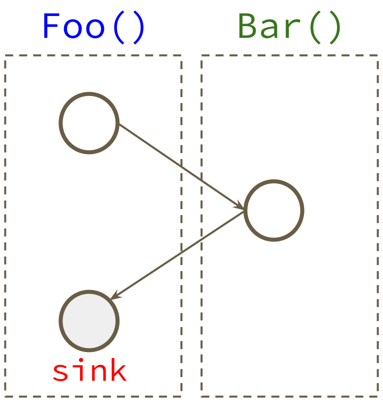
Open window when receiving network command

Runtime Logging

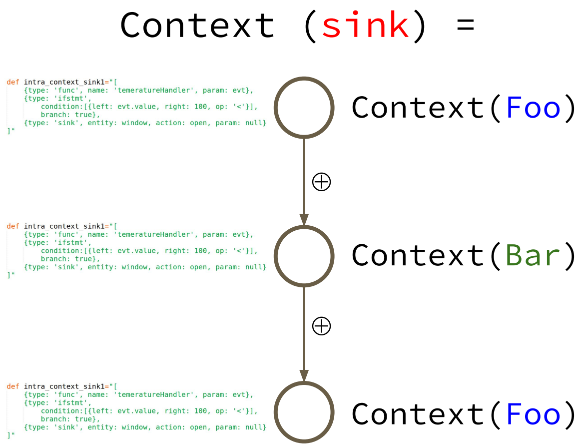
- Dynamic taint tracking for SmartThings' Groovy
  - Data dependency tracking: field-sensitive, implicit taint
    - E.g., luminance as side channel attack
  - Construct runtime context: maintain caller info for each method
    - Merge intra-procedure context based on the call stack



Luminance as side channel malware

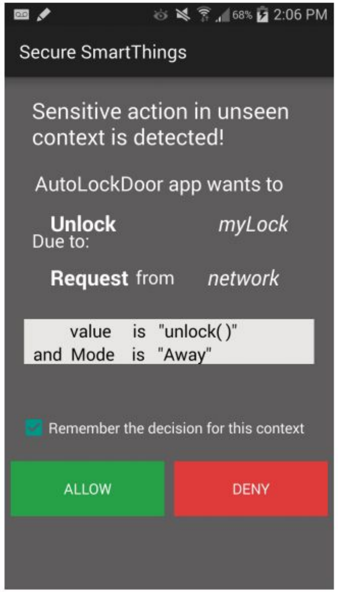


Runtime call stack



End-to-end Implementation

- Patch the app with the context-based permission requesting logic
- A permission server to manage permission granting decisions
- Presentation of context on Android:



Legitimate logic: automatically lock door after 240 seconds

Backdoor logic: unlock the door when receiving network request

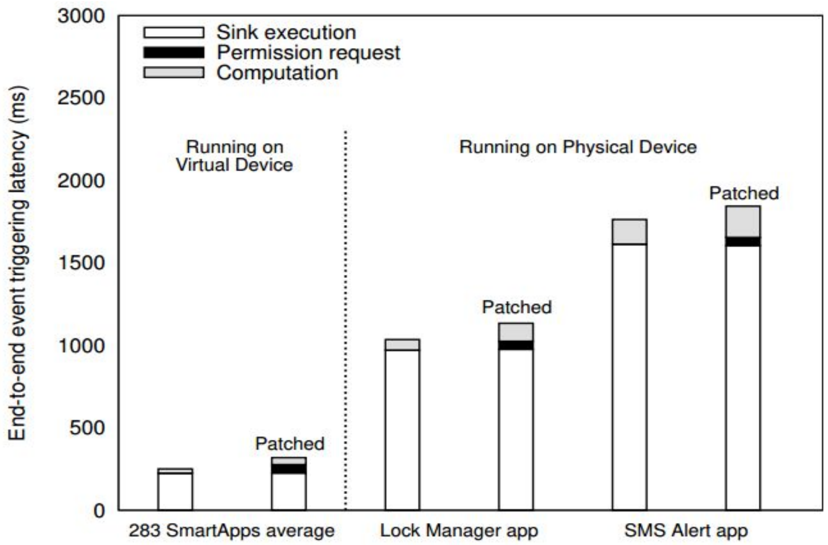
Evaluation

- Effectiveness:
  - Dataset: 283 commodity SmartApps, 25 malicious SmartApps created based on the evasion attack taxonomy, including 3 reported SmartThings malware[1]
  - Result:
    - Sensitive functionalities of all the SmartApps are correctly patched
    - Malicious paths of the 25 malware can be distinguished without ambiguity
- Prompt frequency:
  - Fuzz testing on the 283 commodity SmartApps by injecting events
  - Mean life-time permission prompts are only **3.5**

[1] "Secure Analysis of Emerging Smart Home Applications", *Fernandez et al. Oakland' 16*

Evaluation (Cont'd)

- Performance overhead: latency
  - Tested on both virtual and physical devices
    - +26.7% latency on virtual devices
    - +4.5% ~ 9.6% latency on physical devices



Breakdown of the end-to-end event trigger latency on virtual and physical devices

Conclusion and Future Work

- Future Work:
  - Usability: better presentation of context in IoT scenarios
  - Efficiency: adapt when app functionality is enriched
- Conclusion:
  - A context definition that defeats known classes of malware on appified platforms
  - ***ContexIoT*** approach that help enforce contextual integrity in IoT apps

We released our malware dataset:

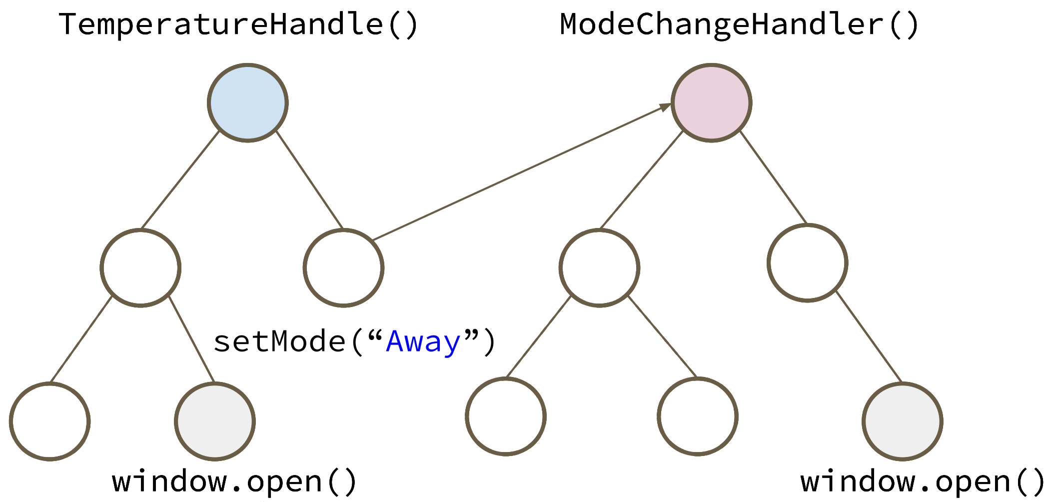
https://tinyurl.com/contexIoT

Conclusion

Backup

Static Analysis: Pruning

- Instrument Groovy AST transformation to build CFG
  - Prune code that are not in the CFG from `Source` to `Sink`
  - Adapt to the trigger-action based programing model
  - Exception: add missing edges to model side-effects



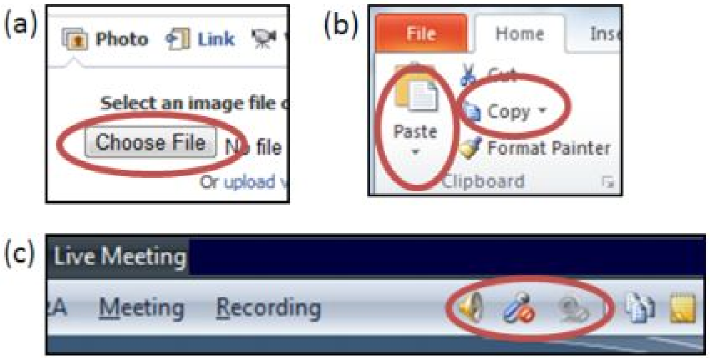
Appification of IoT

- Evolution of software platform:

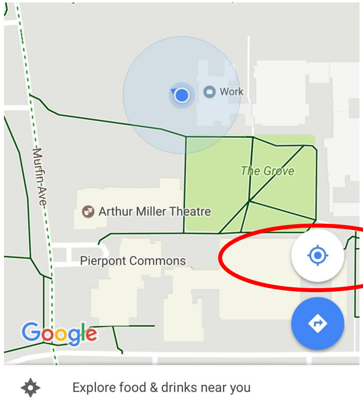
Functional | **Software Defined** | **Applified**

Permission System Revisit: Implementations

- Contemporary implementations:
  - Install time prompt (Android 5.x, SmartThings, ...)
  - Runtime prompt (iOS, OSX, Windows, ...)
  - User-driven access control (research prototype, some apps, ...)



Example Access Control Gadgets



Location button on Google Maps

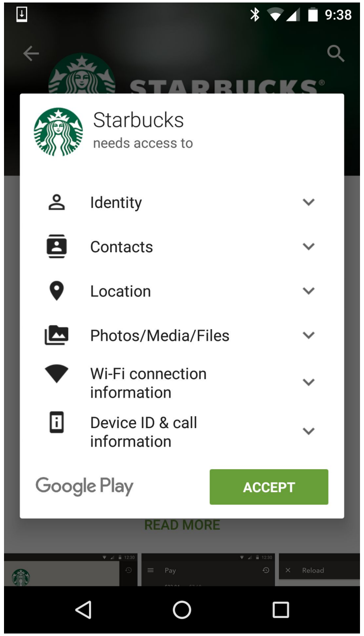
Appified Platform

- Software platform that supports 3rd party app development
- The evolution of software platform

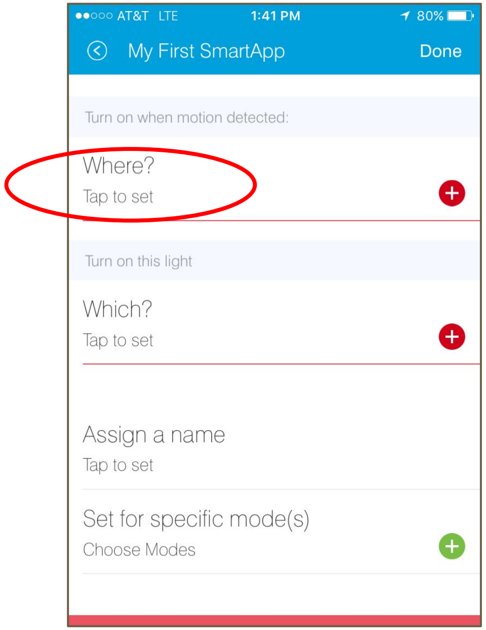
Functional | **Software Defined** | **Appified**

Permission System Revisit: Implementations

- Contemporary implementations:
    - Install time prompt (Android before 5.x, SmartThings, ...)



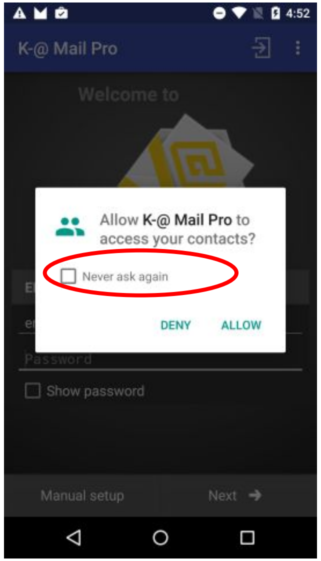
App installation on Android 5.0



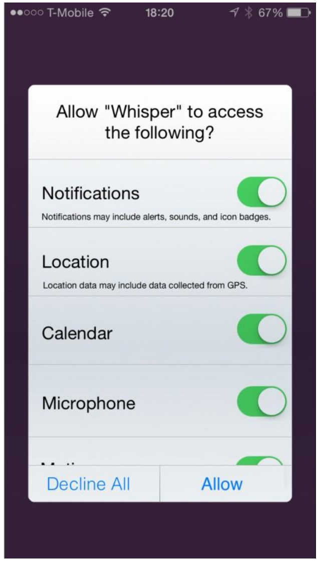
App installation on Samsung SmartThings

Permission System Revisit: Implementations

- Contemporary implementations:
  - Install time prompt (Android 5.x, SmartThings, ...)
  - Runtime prompt (iOS, OSX, Windows, ...)



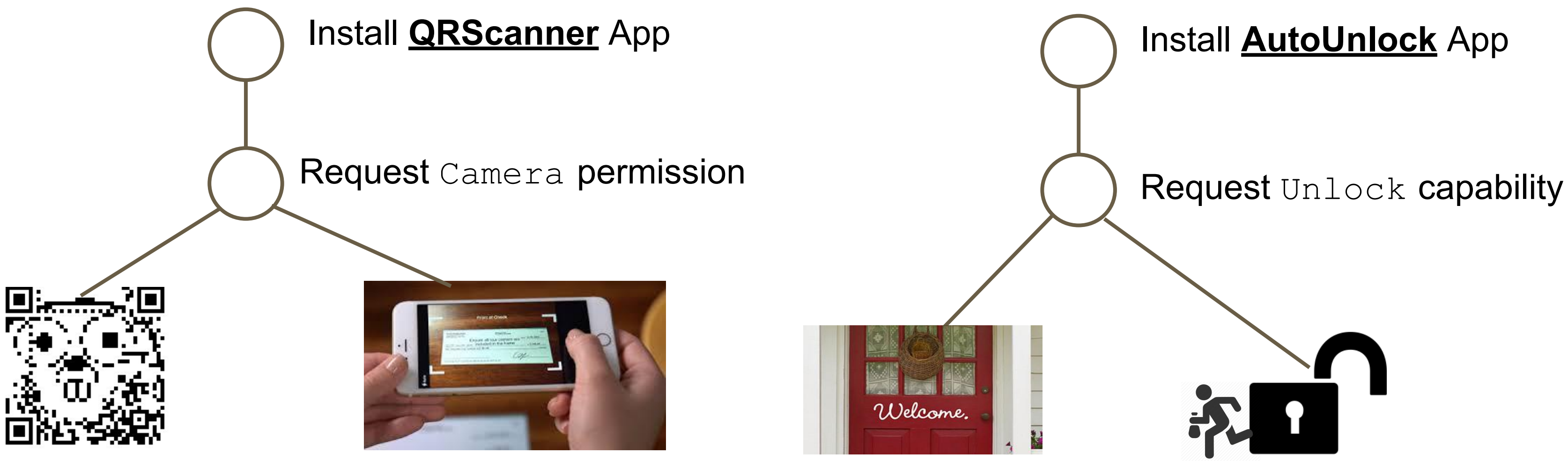
Runtime prompt on Android 6.0



Runtime prompt on iOS 9.x

Permission System Revisit: Limitations

- Installation time and runtime prompt solutions:
    - Coarse-grained, insufficient, undemanding



Ideal access control needs to be In-Context !

Providing Contextual Integrity to IoT Platform

Challenges	*Solutions*
(1) **Definition** of context	[1] Extensive survey of evasion attacks
(2) **Availability** of context	[2] **ContexIoT** to extract context information
(3) **Frequency** of prompts	[3] Adapt context comparison to reduce prompts

Attack Taxonomy Methodology

- Extensive survey of attacks reported on
  - Existing IoT devices (12)
  - Smartphone platform (17)
- Construct misbehave SmartApps that achieve similar malicious functionality (25)
- Evaluate the attacks against representative work in permission evolution/revolution

Context Definition

Name	Description	Definition of context					Decision made in context?
		Uid/Gid	UI Activity	Control flow	Runtime value	Data flow	
ACG [56]	User-driven access control	✓	✓				✓
AppContext* [68]	Static context-based analysis for malware detection	✓		✓		✓	-
AppFence [41]	Protecting private data from being exfiltrated	✓				✓	
Aurasium [67]	Repackaging app to attach policy enforcement code	✓	✓		✓		✓
CRePE [30]	Enforcing context-based fine-grained policy	✓			✓		
FlaskDroid [25]	Fine-grained MAC on middleware and kernel layer	✓			✓		
SEAndroid [60]	Flexible MAC for Android apps	✓					
SEACAT [31]	Integrating both MAC and DAC in the policy checks	✓	✓		✓		✓
TaintDroid [33]	Dynamic taint tracking and analysis system	✓	✓		✓	✓	✓
TriggerScope* [39]	Static trigger-based analysis for malware detection	✓		✓		✓	-
ContexIoT	Providing contextual integrity to permission granting	✓	-	✓	✓	✓	✓

* These work focus on detecting malicious behavior with static analysis, but not enforcing access control at runtime. However, their methodologies of distinguishing benign and malicious behavior are based on their definitions of context.

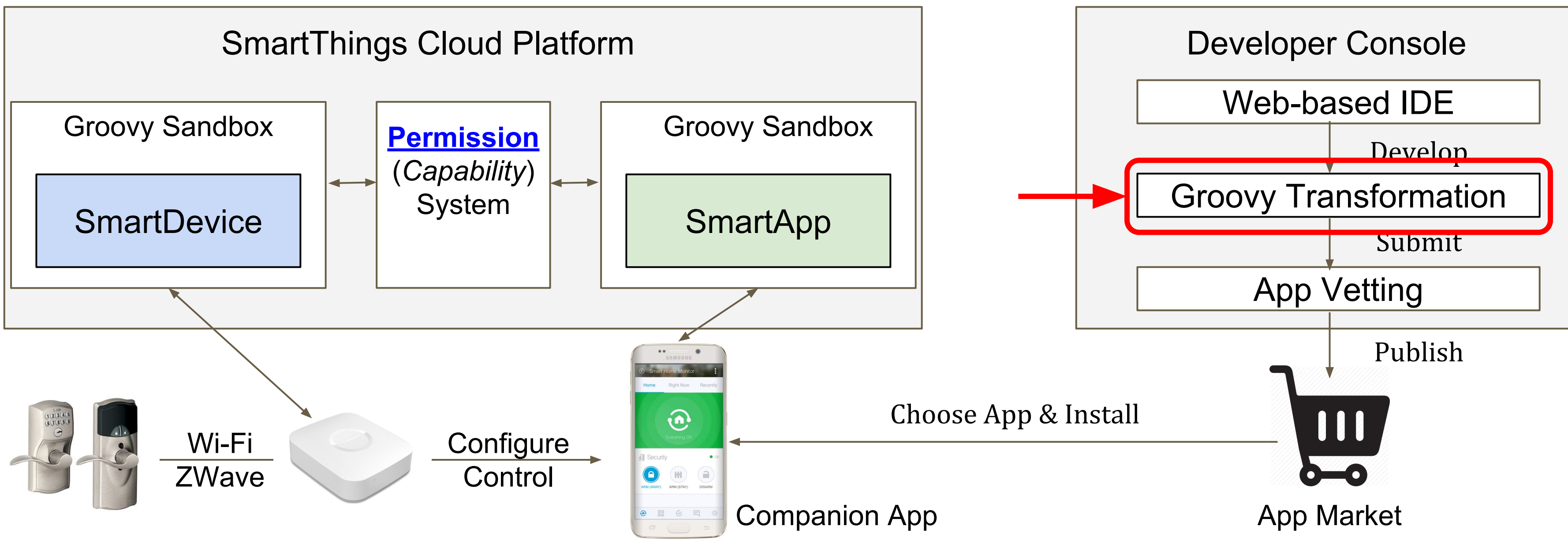
Context definition comparison among representative related work

ContexIoT Design Goal

- **Objective**: *Prompt users with essential context to grant access to desired app behavior*

- Why SmartThings:
  - Relatively mature: 500~ apps, 1
- **Design Challenges**: limited access to the cloud-backed IoT platform

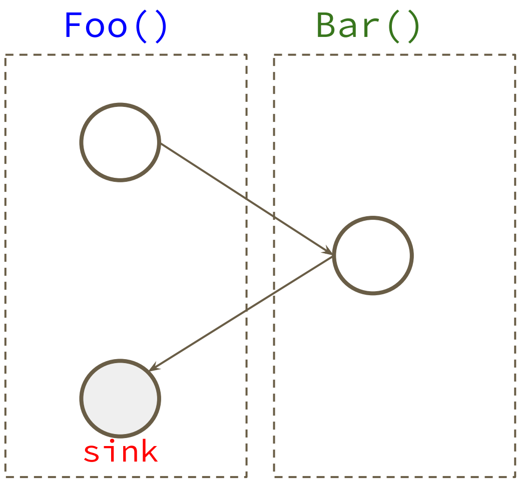
Design: SmartThings Background



Cloud-baked SmartThings Ecosystem

Runtime Logging: Construct Context

- Construct inter-procedural context
  - Reconstruct call stack by maintaining the caller function for each method
  - Construct context by assembling intra-procedure context based on the call stack




```
Context (sink) =
     Context(Foo)
   ⊕ Context(Bar)
   ⊕ Context(Foo)
```

Outline:

- Methodology:
    - Attack & related work taxonomy
        - Extensive survey of evasion attacks to complete the definition of context
    - Design of ***ContexIoT*** on SmartThings platform, which
        - Collects essential context of IoT apps in real time
        - Enforces contextual integrity for each permission granting decisions
    - Evaluation on commodity apps and sample malware