ContexIoT: Towards Providing Contextual Integrity To Appified IoT Platform


University of Michigan
Appified Platform

- Software platform that supports 3rd-party app development

- 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

SmartThings app setup page
Android 6.0 runtime prompt
Permission System: Practices & Limitations

**Installation time prompts:** decision made out-of-context

**Runtime prompts:** coarse-grained and insufficient

---

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.

\[
\text{Context}(T_{using}) = \text{Context}(T_{granted})
\]

● 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

Mobile adversary techniques
- Repackaging
- Shadow payload
- Side channel

Evaluate attacks against context definitions in related work

Context components
- Control flow
- Data flow
- ……
# Context Definition

<table>
<thead>
<tr>
<th>Related work</th>
<th>Context components</th>
<th>Decision made in context?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>UID/GID</td>
<td>UI Activity</td>
</tr>
<tr>
<td>ACG</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>CRePE</td>
<td>✔</td>
<td>✗</td>
</tr>
<tr>
<td>AppContext</td>
<td>✔</td>
<td>✗</td>
</tr>
<tr>
<td>AppFence</td>
<td>✔</td>
<td>✗</td>
</tr>
<tr>
<td>Aurasiun</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>FlaskDroid</td>
<td>✔</td>
<td>✗</td>
</tr>
<tr>
<td>SEAndroid</td>
<td>✔</td>
<td>✗</td>
</tr>
<tr>
<td>SEACAT</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>TaintDroid</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>TriggerScope</td>
<td>✔</td>
<td>✗</td>
</tr>
</tbody>
</table>

**ContexIoT** ✔ N/A ✔ ✔ ✔ ☑ 9
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

SmartThings Cloud Platform

SmartDevice → Permission (Capability) System → SmartApp

User Space

Configured Devices: Wifi, ZWave

Companion App

Developer Console

Web-based IDE

App Instrumentation

App Vetting

Choose App & Install

App Market

Cloud-backed SmartThings Ecosystem
ContextIoT Design

- Patch SmartApps with the access control logic

<table>
<thead>
<tr>
<th>Trigger</th>
<th>Original IoT App Logic</th>
<th>ContextIoT Patched App Logic</th>
<th>ContextIoT Cloud Backend</th>
<th>Companion App</th>
</tr>
</thead>
<tbody>
<tr>
<td>Temperature &gt; 100 °F</td>
<td>window.open()</td>
<td>window.open()</td>
<td>Context 1 (✓)</td>
<td>Prompt to user</td>
</tr>
<tr>
<td>HomeMode == “Sleep”</td>
<td></td>
<td></td>
<td>Context 2 (✗)</td>
<td></td>
</tr>
</tbody>
</table>

Context(\(T_{using}\)) == Context(\(T_{granted}\))?

Reuse() : Prompt()

End-to-end secure logic

Context collection 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

```
Temperature > 100
Mode == "sleep"
```

```
window.open()
```
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

```java
def temperatureHandler(evt)
{
    if(100<evt.value)
    {
        window.open() //sink1
    }
}

def requestHandler(evt)
{
    def command = evt.params
    if("open" == command)
    {
        window.open() //sink2
    }
}
```

```java
def intra_context_sink1="[
    {type: 'func', name: 'temperatureHandler', 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

```java
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

Context (sink) = Context(Foo) ⊕ Context(Bar) ⊕ Context(Foo)
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](image1)
  ![Backdoor logic](image2)

**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
    - TemperatureHandle() ModeChangeHandler()
  - Exception: add missing edges to model side-effects

```
setMode("Away")
window.open()
window.open()
```
Appification of IoT

- Evolution of software platform:
Permission System Revisit: Implementations

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

Example Access Control Gadgets

Location button on Google Maps
Appified Platform

- Software platform that supports 3rd party app development
- The evolution of software platform
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 (Android 6.0, iOS 9.x)
Permission System Revisit: Limitations

Installation time and runtime prompt solutions:

Coarse-grained, insufficient, undemanding

Install **QRScanner** App

Request **Camera** permission

Install **AutoUnlock** App

Request **Unlock** capability

Ideal access control needs to be **In-Context**!
## Providing Contextual Integrity to IoT Platform

<table>
<thead>
<tr>
<th>Challenges</th>
<th>Solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) <strong>Definition</strong></td>
<td>[1] Extensive survey of evasion attacks</td>
</tr>
<tr>
<td>of context</td>
<td></td>
</tr>
<tr>
<td>(2) <strong>Availability</strong></td>
<td>[2] <strong>ContexIoT</strong> to extract context information</td>
</tr>
<tr>
<td>of context</td>
<td></td>
</tr>
<tr>
<td>(3) <strong>Frequency</strong></td>
<td>[3] Adapt context comparison to reduce prompts</td>
</tr>
<tr>
<td>of prompts</td>
<td></td>
</tr>
</tbody>
</table>
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

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

*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

SmartThings Cloud Platform

- Groovy Sandbox
  - SmartDevice
  - Permission (Capability) System
  - Groovy Sandbox
  - SmartApp

Developer Console

- Web-based IDE
- Groovy Transformation
- App Vetting
- App Market

Cloud-baked SmartThings Ecosystem

- Wi-Fi
- ZWave
- Configure Control
- Choose App & Install
- Companion App
- Develop
- Submit
- Publish
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