WILLIAM & MARY
CHARTERED 1693

# CSCI 780: IoT Security

Prof. Adwait Nadkarni

Lecture 3

# Motivation

- Many platforms are now *programmable*
- Developers can use the API to build *apps that*
    - Get status updates from devices
    - Send commands to devices
    - Interface with other services (SMS, Web Services)
- Prior work has looked at: devices, the cloud, the OS

Key question: *Is this API secure?*

# Motivation

Key question: *Is this API secure?*

### Integrity

Can attackers manipulate devices? (e.g., insert lock codes)

### Availability

Can attackers disable devices? (e.g., turn OFF a camera)

### Privacy

Can attackers learn private information? (e.g., the user’s schedule)

### Authenticity

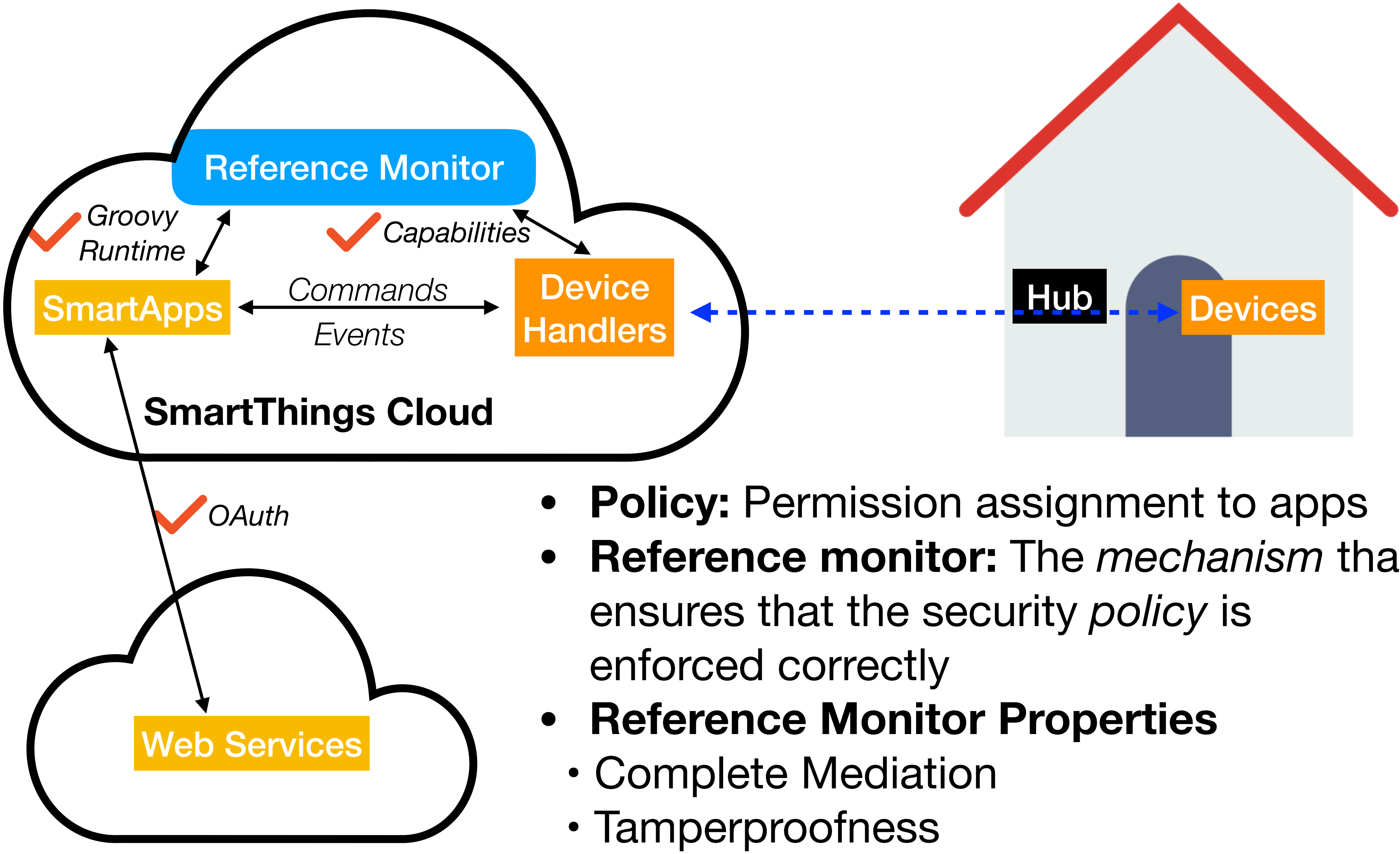
Can attackers spoof messages? (e.g., event spoofing, using stolen OAuth tokens)

### Confidentiality

Can attackers learn sensitive information (e.g., lock codes)

# Background

- SmartThings uses both the hub and the cloud (pre-2019)



- **Policy:** Permission assignment to apps
- **Reference monitor:** The *mechanism* that ensures that the security *policy* is enforced correctly
- **Reference Monitor Properties**
  - Complete Mediation
  - Tamperproofness
  - Verifiability

# Methodology

- Dynamic Testing
- Static Analysis
  - Source code (Groovy SmartApps)
  - Binaries (certain Android apps)
- Network Analysis (mainly to build the dataset)
- **Research Questions:**
  - How *overprivileged* are apps?
  - Can events be *spoofed*?
  - What sensitive information can apps access?
  - How do external third-party integrations affect security?
  - …

# Findings

- Overprivilege
- Event injection (*i.e., spoofing)*
- Event Sniffing
- Vulnerable Third-party integrations

# Findings: Overprivilege

- **Coarse-grained Capabilities** Policy
  - App asks for capability “lock”
    - Can read the lock’s state, and issue the “lock” and “unlock” commands.
  - *What if the app only needs to read the lock state?*
- **Device-granularity binding** Mechanism
  - Apps get *all* capabilities for a device, if they ask for just one.

*Which of these is a policy problem, vs a mechanism problem?*

*Which of these would be harder to fix?*

# Findings: Event Injection

- **Dynamic code loading**
  - SmartApps use dynamic method invocation
  - Can be exploited to execute any code in the SmartApp's *security context (i.e., the capabilities available to the SmartApp)*
- **Event spoofing is trivially possible**
  - Direct Approach: Spoof an event message, with the 128 bit ID of the device
  - Indirect Approach: Modify the *locationMode*. No access control policy protecting it!

# Findings: Sniffing

- **A SmartApp can listen to *everything* from a *bound* device**
  - No access control in place
  - Can subscribe to all events, if binding is established.

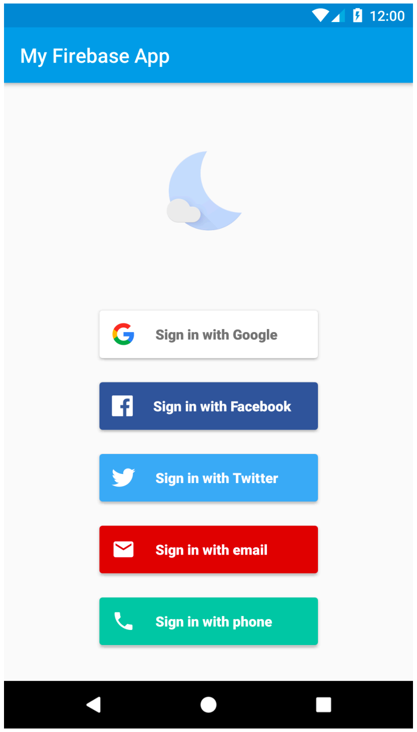
- **A SmartApp can listen to *everything* if it knows the 128 bit *device ID***
  - Even if the device is not bound to the SmartApp

*Why is this bad?*

*How can the adversary get this device ID?*

# Findings: Vulnerable 3rd Party Integrations

- ***OAuth tokens*** **can be stolen, or rather, *falsely acquired***
  - OAuth tokens enable a 3rd-party to connect to the user's SmartThings account.
  - To successfully acquire an OAuth token for a user's SmartThings account, a Web service needs:
    1. a *client ID*
    2. a *client secret*
    3. the user to sign in, and redirect a *code* to the Web service.
  - Mobile apps often hardcode the client ID and secret, and reduce the barriers to acquiring a token.



# Attack!

1. Inject Key Codes!
    1. Acquire (Steal) Token + Inject Commands (using capabilities not requested)
2. Pin Code Snooping:
    1. Acquire device ID or bind to the device (e.g., battery monitor) + register for certain events (e.g., CodeReport)
3. Disabling Vacation Mode
4. Fake Alarm

# Suggestions/Discussion

- *Risk-based capabilities* would prevent overprivilege.
  - User-studies to quantify risk
- *A unified security perspective across platforms* (mobile and smart home) to identify the impact of vulnerable integrations
- *App Identity* to prevent event spoofing
  - cryptographic?
  - Android intents? (i.e., each app has a UID that is carried by its processes)
    - What about spoofing from devices?

# Takeaway

- New platforms are *very* vulnerable
- Third-party integrations may weaken security

# Questions!

1. How do we make capabilities finer-grained? Is more user-studies the correct approach (i.e., *to quantify risk*)?
2. What are the problems with the *device-binding* model?
3. "Programming frameworks are difficult to change without significant disruption once there is a large set of applications…" —> DISCUSS
4. How can we obtain *automation with security* (i.e., as security policies may disrupt automation)
5. How necessary/useful was the *user survey*?
6. What does this mean for the *apps we use*?