SECURITY ANALYSIS OF EMERGING SMART HOME APPLICATIONS

Earlence Fernandes, Jaeyeon Jung, Atul Prakash

Presented by
Surya Mani
Content

- Motivation
- Related Work
- SmartThings-Big Picture
- Security Analysis
- Proof-of-concept attacks
- Defense Mechanism
Motivation

- Huge number of connected gadgets, systems and appliances that do a wide variety of different things.
- Though it provides user with benefits, it also expose user to security risks
Related Work

- A framework for evaluating security risks associated with technologies used at home - Denning

- Device front
  - MyQ garage system, Wink Relay touch controller, Honeywell Tuxedo Touch Controller
  - Investigate the feasibility of causing physical harm through the explosion of CFLs through an exploited home automation system
  - Use Case: sharing smart devices with others

- Protocol Front – Zigbee and Zwave protocol

- Investigation on cause of over privilege due to insufficient API documentation and guidelines on different types of permission - Felt
First in-depth security analysis of one such “smart home” platform that allows anyone to control their home appliances from light bulbs to locks with a PC or smartphone.

Demonstrate programming framework design flaws

Analyze protocol operating between SmartThings backend and the client-side web IDE

Remote attacks that weaken the home security system independent of specific protocol in use.

Evaluation of SmartThings capability model in protecting sensitive device operations
Smart Home applications
SmartThings

- SmartThings interconnects separately operating home appliances to create a fully connected SmartThings home controlled by smartphone apps.
- The main goal of SmartThings is to provide a new class of automation by connecting appliances to one another, to the Internet, and to homeowners.
Big Picture
SmartThings - cont.

Three main components
- Hubs
- SmartThings Cloud Backend
- Smartphone companion app
SmartThings System

- SmartApps and SmartDevices
- Capabilities and Authorization
- Events and Subscriptions
- Webservice SmartApps
- Sandboxing
SmartApp Structure

```python
definition(
    name: "DemoApp", namespace: "com.testing",
    author: "IoTPaper", description: "Test App",
    category: "Utility")

// query the user for capabilities
preferences {
  section("Select Devices") {
    input "lock1", "capability.lock", title: "Select a lock"
    input "swl", "capability.switch", title: "Select a switch"
  }
}
def updated() {
  unsubscribe()
  initialize()
}
def installed() {
  subscribe swl, "switch.on", onHandler
  subscribe swl, "switch.off", offHandler
}

def onHandler(evt) {
  lock1.unlock()
}

def offHandler(evt) {
  lock1.lock()
}
```

Listing 1. SmartApp structure.

Fig. 2. Installation user interface and device enumeration: This example shows that an app asks for devices that support capability.lock and capability.switch. The screen on the right results when the user taps on the first input field of the screen on the left. SmartThings enumerates all lock devices (there is only one in the example). The user must choose one or more devices that the app can access.
Security Analysis

- Occurrence of over privilege in SmartApps
- Insufficient sensitive event data protection
- Insecurity of third party integration
- Unsafe use of groovy dynamic method invocation
- Unrestricted Communication abilities via API Access control
Occurrence of over privilege in SmartApps

Because of SmartThings Framework

- Capabilities – Coarse-grained, providing access to multiple commands and attributes for a device (55%)
  E.g. Capability.lock (Commands: lock and unlock, attribute: lock)
- SmartApp obtain more capabilities than it request because of SmartApp-SmartDevice binding (42%)
  E.g. SmartApp uses capability.battery
Light
Allow for the control of a light device

Preferences Reference
capability.light  //consider it for Oven

Attributes
switch: ENUM
A string representation of whether the light is on or off
off
The value of the switch attribute if the light is off
on
The value of the switch attribute if the light is on

Commands
 off()
Turn a light off
 on()
Turn a light on

Lock
Allow for the control of a lock device

Preferences Reference
capability.lock

Attributes
lock: ENUM
The state of the lock device
locked
The device is locked
unknown
The state of the device is unknown
unlocked
The device is unlocked
unlocked with timeout
The device is unlocked with a timeout

Commands
 lock()
Lock the device
 unlock()
Unlock the device
Example of over privilege

Fig. 3. SmartApps vs. SmartDevices vs. Physical Devices: When a user installs this SmartApp, SmartThings will show the lock and the motion sensor since both the corresponding device handlers (SmartDevice1 and SmartDevice2) expose the requested capability.
Insufficient sensitive event data protection

Because of insecure event sub-system design

- After a SmartApp is approved to access a SmartDevice, it monitors any data published by SmartDevice (e.g. Lock codes)
- SmartApp which acquired 128-bit identifier (unique to SmartDevice) can monitor all the events.
  
  
  subscribe(deviceObj, attrstring, handler)

- Events generated from devices can be spoofed. As the framework,
  
  - does not have control over raising events
  - verify the integrity or the origin of an event by triggered SmartApps
Insecurity of third party integration

- OAuth bearer token – attached to request while invoking the WebService SmartApp HTTP endpoints
Unsafe use of groovy dynamic method invocation

- String representation of a command is received over HTTP
  ```groovy
def str = "foo"
```
- The string is executed directly by dynamic method invocation (method can be invoked using name as a string)
  ```groovy
foo()
```

Unrestricted Communication abilities via API Access control

- No restrictions on outbound Internet communication of SmartApps leaks sensitive information
Empirical security analysis

### TABLE II
**BREAKDOWN OF OUR SMARTAPP AND SMARTDEVICE DATASET**

<table>
<thead>
<tr>
<th></th>
<th>Total # of SmartDevices</th>
<th># of device handlers raising events using <code>createEvent</code> and <code>sendEvent</code>. Such events can be snooped on by SmartApps.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>132</td>
<td>111</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Total # of SmartApps</th>
<th>499</th>
</tr>
</thead>
<tbody>
<tr>
<td># of apps using potentially unsafe Groovy dynamic method invocation</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td># of OAuth-enabled apps, whose security depends on correct implementation of the OAuth protocol</td>
<td>27</td>
<td></td>
</tr>
<tr>
<td># of apps using unrestricted SMS APIs</td>
<td>131</td>
<td></td>
</tr>
<tr>
<td># of apps using unrestricted Internet APIs</td>
<td>36</td>
<td></td>
</tr>
</tbody>
</table>

### TABLE III
**COMMANDS/ATTRIBUTES OF 64 SMARTTHINGS CAPABILITIES**

<table>
<thead>
<tr>
<th></th>
<th>Documented</th>
<th>Completed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Commands</td>
<td>66</td>
<td>93</td>
</tr>
<tr>
<td>Attributes</td>
<td>60</td>
<td>85</td>
</tr>
</tbody>
</table>

### TABLE IV
**OVERPRIVILEGE ANALYSIS SUMMARY**

<table>
<thead>
<tr>
<th>Reason for Overprivilege</th>
<th># of Apps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coarse-grained capability</td>
<td>276 (55%)</td>
</tr>
<tr>
<td>Coarse SmartApp-SmartDevice binding</td>
<td>213 (43%)</td>
</tr>
</tbody>
</table>
PROOF-OF-CONCEPT ATTACKS
A. Backdoor pin Code Injection Attack

- Over privilege using SmartApp-SmartDevice coarse-binding
- Stealing an OAuth token using the hard-coded secret in the existing binary
- Getting a victim to click on a link pointing to the SmartThings Web site
- Command injection to an existing Webservice SmartApp
Stealing the OAuth Token

GET https://graph.api.smartthings.com/oauth/authorize?

- response_type=code&
- client_id=YOUR-SMARTAPP-CLIENT-ID&
- scope=app&
- redirect_uri=YOUR-SERVER-URI

<table>
<thead>
<tr>
<th>parameter</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>response_type</td>
<td>Use code to obtain the authorization code.</td>
</tr>
<tr>
<td>client_id</td>
<td>The OAuth client ID of the SmartApp.</td>
</tr>
<tr>
<td>scope</td>
<td>This should always be “app” for this authorization flow.</td>
</tr>
<tr>
<td>redirect_uri</td>
<td>The URI of your server that will receive the authorization code.</td>
</tr>
</tbody>
</table>
Command Injection Attacks

- WebService SmartApp associated with the third-party Android app uses Groovy dynamic method invocation
- Format of the command string needed to activate the SmartApp endpoint

Listing 2: Portion of the Logitech Harmony WebService SmartApp available in source form. The mappings section lists all endpoints. Lines 19 and 21 make unsafe use of Groovy dynamic method invocation, making the app vulnerable to command injection attacks. Line 23 returns a HTTP 204 if the command is executed. Our proof-of-concept exploits a similar WebService SmartApp.
B. Door Lock Pin Code Snooping Attack

1 zw device:02,
2 command:9881,
3 payload:00 63 03 04 01 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A
4 parsed to
5 [['name': 'codeReport', 'value': 4,
6 'data': {'code': '8877'},
7 'descriptionText': 'ZWave Schlage Lock code 4 set',
8 'displayed': true,
9 'isStateChange': true,
10 'linkText': 'ZWave Schlage Lock']]

---

Fig. 5. Snooping on Schlage lock pin-codes as they are created. We use the Schlage FE599 lock in our tests.
C. Disabling Vacation Mode Attack

- Depends on the “mode” property of the location object
- SmartThings does not have security controls around the SendLocationEvent API
- Even spoofing by the attack SmartApp
- Attack launched from any SmartApp without requiring the specific capabilities
D. Fake Alarm Attack

- Attack launched from any SmartApp without requiring the specific capabilities
- Attack SmartApp is installed in the system
- Even spoofing by the attack SmartApp
- Controlling the device
Survey Study of SmartThings Users
| Table VI

**SURVEY RESPONSES OF 22 SMARTTHINGS USERS**

<table>
<thead>
<tr>
<th>Interest in installing battery monitor SmartApp:</th>
<th>17 77%</th>
<th>4 18%</th>
<th>1 5%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interested or very interested</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Neutral</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Not interested at all</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Set of devices that participants would like the battery monitor app to monitor:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selected motion Sensor</td>
</tr>
<tr>
<td>Selected Schlage door lock</td>
</tr>
<tr>
<td>Selected presence Sensor</td>
</tr>
<tr>
<td>Selected FortressZ alarm</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Participants' understanding of security risks—# of participants who think the battery monitor app can perform the following:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cause FortressZ alarm to beep occasionally</td>
</tr>
<tr>
<td>Send battery levels to remote server</td>
</tr>
<tr>
<td>Send motion and presence sensor data to remote server</td>
</tr>
<tr>
<td>Disable FortressZ alarm</td>
</tr>
<tr>
<td>Send spam email from hub</td>
</tr>
<tr>
<td>Download illegal material using hub</td>
</tr>
<tr>
<td>Send door access codes to remote server</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Participants' reported feelings if the battery monitor app sent out door lock pin codes to a remote server:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upset or very upset</td>
</tr>
</tbody>
</table>
Defense Mechanism
THANK YOU