Embedded Linux Conference: OIC Security Model and Vision

Ned Smith
Intel
Day-in-the-Life Scenario
Security Objectives

• Crossing domain boundaries

• Ad-hoc introductions

• Ensuring access

• Establishing ownership
OIC Terminology

- A **Device** is an OIC stack instance.
- Devices implement roles: **Client**, **Server**, **Intermediary**.
- Devices have **Resources** and perform **Actions**.
- Resources have **Properties**.

**Example Grouping:**

- **Sensor**
  - OIC Server
    - Access Control
    - Resources
- **Controller**
  - OIC Client
    - Actions
- **Actuator**
  - OIC Server
    - Access Control
    - Resources
To Cross a Boundary We Must Define the Endpoint

- An OIC *device* is the endpoint
- ...more specifically it is the OIC resource layer
- OIC resources define how device capabilities are exposed to other OIC devices
- Resources are accessed securely through a secure channel such as DTLS
  - End-to-end message encryption, integrity and replay protection
- OIC does not define endpoint hardening techniques
  - Resource layer hardening is implied
Key Management Objectives

Pair-wise Keys

Group Symmetric Keys

Dynamic Provisioning of Keys

Localized Autonomy

Key Distribution
Ad-hoc Introduction

- Ad-hoc interactions suppose there isn’t a trusted key distributor.

How do they establish pairwise keys?

Out-of-band exchange of a PIN using a smart device with a user interface!

Domain crossings may enlist key management services.
Ensuring Access with Access Control

- Anticipate intended interactions
- Add friction to unintended interactions

- Seamless remote access
How To Distinguish Intended vs. Unintended?

- Access control granularity has four scoping levels
  - Group, Device, Resource and Attribute
- OIC Client actions capture interaction patterns
  - Peer-peer, Observer, Subscribe-notify, etc...
  - Actions specify intended device interactions

Example

Device 1

GET implies READ access

Device 2

Informs

Get:

responses:

GET implies READ access

Collection href identifies which device and resource should have read access
- Access is blocked if no ACL match is found
- Device1 request to get /oic/d is **accepted** due to ACL Read permission
- Device2 request to update /oic/light/1 is **denied** due to time-of-day policy
- An intermediary (Device4) may also enforce ACLs
Property-level Access

Example Resource Definitions:

Without Property Level Access Control

```json
{"$schema": "http://json-schemas.org/schema#",
"id": "http://openinterconnect.org/oic.thing#",
"definitions": {
  "oic.thing": {
    "type": "object",
    "properties": {
      "Property-1": {"type": "type1"},
      "Property-2": {"type": "type2"}
    }
  }
}
```

Properties are opaque to OIC framework

With Property Level Access Control

```json
{"$schema": "http://json-schemas.org/schema#",
"id": "http://openinterconnect.org/oic.thing#",
"definitions": {
  "oic.RsrcProp-1": {
    "type": "object",
    "properties": {
      "Property-1": {"type": "type1"}
    }
  },
  "oic.RsrcProp-2": {
    "type": "object",
    "properties": {
      "Property-2": {"type": "type2"}
    }
  }
}
```

Resources with property-level granularity are NOT opaque to OIC stack

- Per property access can be achieve using a collection resource
  - A new resource is defined containing a single property
- Resource level access mechanism can satisfy property level access requirements
Remote Access

• OIC communications layer accommodates remote access

• Much of the remote access complexity is hidden within the OIC communication abstraction layer

• Home devices use same credentials when outside

• RA services provide a meeting place in the cloud
  – Using user credentials common to cloud services
## Current Techniques for Device Ownership

<table>
<thead>
<tr>
<th>Just Works</th>
<th>Mode Switch</th>
<th>Random PIN</th>
<th>Pre-provisioned PIN</th>
<th>Pre-provisioned Credential</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Just Works" /></td>
<td><img src="image2.png" alt="Mode Switch" /></td>
<td><img src="image3.png" alt="Random PIN" /></td>
<td><img src="image4.png" alt="Pre-provisioned PIN" /></td>
<td><img src="image5.png" alt="Pre-provisioned Credential" /></td>
</tr>
</tbody>
</table>

- Several techniques are in practice today, all impact manufacturing
- Problem is there is a disparity in security and attack vulnerability
- OIC members are working to standardize methods for device owner transfer

Device Owner Transfer Objectives

- Manufacturer supports secure over-the-air transfer of owner

- Agreement that both parties intend to transfer ownership

- Identify the domain in which the new device is transferred

- Trust in the endpoint device performing the transfer steps
Example Device Owner Transfer Protocol

- Manufacturer PIN exchanged out-of-band authorizes intent
- Device certificate identifies owner group / domain
- Attestation of device internals to ensure trusted operation
- Diffie-Hellman for secure ad-hoc exchange of protocol messages
Conclusion

• IoT use models demand strong but flexible security
  - Devices operate in autonomous and ad-hoc ways
• OIC key management supports end-to-end device protection
• Devices from different domains can establish ad-hoc pair-wise keys using Diffie-Hellman
• Resource layer ACLS allow intended interactions while preventing unintended interactions
• Secure device ownership helps prevent attacks when devices are added to the network
Call to Action

• OIC is working toward interoperable IoT security
• Your participation in OIC is the best way to ensure your IoT products interoperate securely
Questions?

Ned M. Smith
ned.smith@intel.com
openinterconnect.org