Authentication and Authorization for IoT Devices in Disadvantaged Environments

Sebastian Echeverria
Copyright 2019 Carnegie Mellon University. All Rights Reserved.

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

This material may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.


DM19-0279
Motivation

• IoT devices are increasingly being used in disadvantaged environments — issues include:
  • Disconnected, intermittent, and limited network connectivity
  • Threats of sabotage, capture, and impersonation
• Authentication and Authorization for Constrained Environments (ACE)
  • IETF draft standard based on OAuth 2.0
  • Adapts OAuth to limitations of IoT devices (e.g., no user intervention, limited resources)
  • Not specifically for disadvantaged environments
OAuth 2.0/ACE Authorization Flow

1. Token Request
2. Access Token + Access Information
3. Token + Request
4. Protected Resource
ACE Limitations

- Bootstrapping is intentionally out of scope

- But is needed in disadvantaged environments due to capture and impersonation risks

- Token revocation is not addressed (other than time-based expiration)

- Notifying devices of compromised nodes is critical in disadvantaged scenarios
Our Work

• Evaluate, adapt, and implement a system based on ACE

• Extend it to address bootstrapping and token revocation for disadvantaged environments

• Evaluate resilience and behavior
Authentication and Authorization for Constrained Environments Framework (ACE)

Introduction > Approach > Implementation > Results

(1) Token Request

Client (C) → Authorization Server (AS)

(2) Access Token + Access Information

(3) Token + Request

(4) Protected Resource

Authorization Server (AS) → Resource Server (RS – IoT Device)

Uses alternative technologies that require less resources
• HTTP → CoAP
• JSON → CBOR
• JOSE → COSE

Eliminates need for consistent online access to an AS through self-contained, proof-of-possession (PoP) tokens associated to cryptographic keys

Tokens can only be decrypted by RS

PoP keys allow clients to prove a token was issued to them

Eliminates need for consistent online access to an AS through self-contained, proof-of-possession (PoP) tokens associated to cryptographic keys
Extensions to ACE – Bootstrapping

• Bootstrapping
  • Grant clients and RSs access to AS services
  • Exchange credentials to secure communications and tokens

• Assumptions:
  • AS in the field may be disconnected from the enterprise
  • IoT devices (RS) may have no input/output devices

• **Approach:** use credentials stored on a printed QR code on IoT devices to link devices through a pairing procedure
## Extensions to ACE – Bootstrapping Alternatives

<table>
<thead>
<tr>
<th>When</th>
<th>Details</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>At manufacturing time</td>
<td>QR code has public key, private key embedded in hardware</td>
<td>Good for addressing device impersonation</td>
<td>IoT devices needs to support hardware-based credentials</td>
</tr>
</tbody>
</table>
| Before deployment     | QR code has credentials read by AS before deployment to the field       | - Out-of-the-box deployment in the field  
- QR code does not have to stay with device                                                              | Loss of flexibility (more devices can’t be added in the field)                                          |
| During deployment     | QR code has PSK read by AS in the field and used for secure ACE connections | Pairing can be done in the field  
- ACE PSK is available to anyone with access to QR code,  
- PSK remains constant between deployments                                                              | IoT device needs to support storing new credentials                                                     |
| During deployment     | QR code has PSK used to secure exchange of second PSK, used for secure ACE connections | - Pairing can be done in the field  
- ACE PSK is different for each deployment                                                                 |                                                                                                          |
Extensions to ACE – Bootstrapping – Pairing Process
Extensions to ACE – Token Revocation

- If a Client or RS is compromised:
  - Clients will have access to resources until their token expires
  - RS can report fake information to clients with valid tokens for them

- Short expiration times are not appropriate as Client or RS may be out of range of AS for long periods of time

- **Approach**: Opportunistic introspection requests from Clients and RSs to AS
  - Whenever they are in range of each other
Extensions to ACE – Token Revocation

**Diagram Description**

- **Client**
  - DTLS Session
  - C-ID
  - Loop
    - For All Active Tokens
      - Introspection Request (EncryptedToken)

- **Authorization Server**
  - RS-ID = GetKeyID(EncryptedToken)
  - PSK_R5 = GetKey(RS-ID)
  - Token = Decrypt(EncryptedToken, PSK_R5)
  - Validate Token (Token)
  - Active / Inactive
  - If Invalid
    - Remove Token (EncryptedToken)
Threat Model – Pairing Procedure and Resource Access
## Threat Model – Pairing Procedure

<table>
<thead>
<tr>
<th>Threat</th>
<th>Mitigation</th>
</tr>
</thead>
</table>
| Node impersonation                             | - RS (IoT device) can only be paired once without rebooting  
- Rebooting and QR code should be physically protected                                                |
| Data flow sniffing                             | - PSK in QR code is used to encrypt communication using DTLS                                                                                   |
| Data flow tampering                            | - PSK in QR code is used to encrypt communication using DTLS                                                                                   |
| Unauthorized credential store access           | - In RS, no stored credentials exist before pairing procedure  
- QR code should be physically protected from view in the field                                                                                  |
| Execution of unauthorized code                 | - Pairing protocol is simple and sanitizes input to avoid arbitrary code execution                                                           |
### Threat Model – Resource Access

<table>
<thead>
<tr>
<th>Threat</th>
<th>Mitigation</th>
</tr>
</thead>
</table>
| Node impersonation                          | - ACE is used along with DTLS and PoP keys to ensure authentication  
- Token revocation can be used if node has been compromised                                                                                   |
| Data flow sniffing                         | - DTLS connections provide integrity protection                                                                                                                                                         |
| Data flow tampering                        | - DTLS connections provide traffic encryption                                                                                                                                                             |
| Unauthorized credential store access       | - AS stores can be encrypted  
- RS stores would need hardware solutions to be better protected                                                                                   |
| Execution of unauthorized code             | - Resource access protocol is simple and sanitizes input to avoid arbitrary code execution                                                                                                               |
| Elevation using impersonation              | - Since OAuth assumes AS is trusted, there is no good mitigation for AS issuing tokens to itself  
- Tokens are separate for each RS, to avoid that a compromised RS impersonates as a client to other RSs                                                                                               |
Architecture – Components
Architecture – Code and Libraries

ACE Authorization Server (Java)
  - Server
  - ACE-Java
  - COSE-Java
  - Scandium (DTLS)
  - CBOR-Java
  - Californium (CoAP)
  - Java JRE

ACE Resource Server – Constrained Version (C)
  - RS Server and COSE parsing
  - TinyDTLS
  - Cn-cbor
  - Erbium CoAP Server
  - Contiki OS (6lbr fork)

Legend
- System or Subsystem Boundary
- Custom Module
- 3rd Party Library
- 3rd Party Lib.
Architecture – Devices and Networks

Client (Laptop)
- ACE Client
- Wi-Fi NIC
- USB Port

Wi-Fi Access Point

IoT Device (CC2538EM)
- ACE Resource Server
- 802.15.4 NIC
- 802.15.4 RF Dongle (CC2531EMK)

Authorization Host (PC)
- Ethernet NIC
- ACE Authorization Server
- USB Port

802.15.4 RF Dongle (CC2531EMK)

Legend
- Hardware Device
- Custom Runtime Component
- Hardware Component

Communication types:
- Ethernet
- Wi-Fi
- 802.15.4
- USB
- IPv6
## Resource Consumption

<table>
<thead>
<tr>
<th>Platform</th>
<th>Min Binary (bytes)</th>
<th>ACE Binary (bytes)</th>
<th>Increase (%)</th>
<th>ACE RAM Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raspberry Pi</td>
<td>995,783</td>
<td>1,452,153</td>
<td>45.83</td>
<td>2.3-3.5 MB</td>
</tr>
<tr>
<td>TI CC2538EM</td>
<td>286,660</td>
<td>354,432</td>
<td>23.64</td>
<td>&lt; 32 KB</td>
</tr>
</tbody>
</table>
Vulnerability Analysis – Attack Graph
Conclusions and Further Work

• System was implemented and tested, and it mitigates most threats
  • While still being ACE-compliant

• Code currently available in Github
  • https://github.com/SEI-TTG/ace-client/wiki

• Working on reducing resource consumption of constrained RS

• Further work: Create a mesh of devices to route introspection requests through them for token revocation checks
Contact Information

**Presenter**

Sebastián Echeverría (SEI - SSD/TTG)
Software Engineer
Email: secheverria@sei.cmu.edu
Telephone: +1-412-268-6552

**Team Members**

Grace A. Lewis (SEI - SSD/TTG)
Dan Klinedinst (ShieldAI – Formerly SEI - CERT/VUL)
Ludwig Seitz (SICS)