draft-ietf-6tisch-minimal-security

Mališa Vučinić, University of Montenegro
Jonathan Simon, Analog Devices
Kris Pister, UC Berkeley
Michael Richardson, Sandelman Software Works
Status

• News
  • draft-ietf-6tisch-minimal-security-04
  • Published on Oct 30th 2017
  • Relies completely on PSKs
  • Summary of updates in -04
Update #1: Key/Nonce Derivation

• OSCORE-06 (formerly known as OSCOAP) updated the key/nonce derivation process
  • Same nonce used for both request and response but under a different key
• We could no longer use “EUI-64 | 0x00” and “EUI-64 | 0x01” as identifiers for the pledge and the JRC
• We now use EUI-64 of the pledge as Master Salt during key derivation and transport it as Context Hint
  • Sender ID of each pledge is 0x00; ID of the JRC is 0x01
Derivation of Key Used to Protect Join Request

- **PSK**
- **Master Salt** = Pledge's EUI-64
- **String “Key”**
- **Algorithm ID and key len.**
- **Sender ID** = 0x00

**HKDF**

"Join Request" Key
Derivation of Key Used to Protect Join Response

PSK

Master Salt = Pledge’s EUI-64

String “Key”

Algorithm ID and key len.

Sender ID = 0x01

HKDF

“Join Response” Key
Nonce Derivation (used both for Join Request and Response)

- PSK
- Master Salt = Pledge's EUI-64
- String “IV”
- Algorithm ID and IV len.

HKDF

Common IV

Nonce
Update #2: Error Handling

• Error handling in -03 opens the pledge to a DoS attack
  • Attacker could send (unprotected) error messages and force the pledge to attempt joining the next advertised network

• Solution in -04:
  • Using Non-Confirmable CoAP msg for Join Request will make OSCORE at JRC silently drop the request in case of failure (decryption, replay, unauthorized)
  • The pledge MUST silently discard any response not protected with OSCORE, including error codes.
  • Forces the pledge to implement a retransmission mechanism at the APP layer duplicating CoAP Confirmable msg functionality
Update #3: Join Request Retransmissions

• Binary exponential back-off mechanism to be implemented by the pledge at the APP layer specified in -04:
  • Super simple, inspired by the one in RFC7252 (CoAP)
  • Pledge keeps track of timeout and retransmission_counter
  • Parameters: TIMEOUT, TIMEOUTRANDOM_FACTOR, MAX_RETRANSMIT
  • *If the retransmission counter reaches MAX_RETRANSMIT on a timeout, the pledge SHOULD attempt to join the next advertised 6TiSCH network.*

<table>
<thead>
<tr>
<th>Name</th>
<th>Default Value</th>
<th>1st attempt: timeout in [10s, 15s]</th>
<th>2nd attempt: timeout in [20s, 30s]</th>
<th>3rd attempt: timeout in [40s, 60s]</th>
<th>4th attempt: timeout in [80s, 120s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIMEOUT</td>
<td>10 s</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIMEOUTRANDOM_FACTOR</td>
<td>1.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAX_RETRANSMIT</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Misc updates

• Recommendation to store untrusted neighbor entries in a separate cache
• Join Request switched from GET -> POST to be more flexible with payload
• Added requirement on persistency of mutable OSCORE context parameters
  • Prevents nonce reuse and replay attacks across reboots
• Extensive editorial pass
  • Rewrote intro, clarifications on the PSK, etc...
Conclusion

• minimal-security-04 relies completely on PSKs
• Tracking of OSCORE, updates to error handling, editorial
• Open issue:
  • Join traffic, potentially controlled by the attacker, can influence SF to trigger 6P commands
  • In minimal-security, we recommend bandwidth cap at Join Proxy but this does not completely solve the problem
  • Should each SF specify how it handles the join traffic? How does one differentiate frames containing Join Requests from other network traffic?
• Reviews welcome!