Implementing and Evaluating IOAM Integrity Protection
https://github.com/iurmanj/ioam-integrity-linux-kernel

Justin Iurman, Benoit Donnet
<firstname.name@uliege.be>

ANRW’24, IETF 120 (Vancouver)
July 23, 2024
IOAM example

IPv6 Packet

Operations:
1. pre-allocate space
2. add IOAM data

Operations:
1. add IOAM data

Operations:
1. add IOAM data

Operations:
1. decapsulate IOAM IPv6 Packet

H P

A

B

C

D

IOAM header
pre-allocated

IOAM header
pre-allocated

IOAM header
pre-allocated
IOAM with Integrity Protection

Operations:
1. pre-allocate space
2. add IOAM data
3. init ICV

Operations:
1. add IOAM data
2. update ICV

Operations:
1. add IOAM data
2. update ICV

Operations:
1. validate ICV
2. decapsulate IOAM IPv6 Packet
IOAM Integrity Protection Header
Option 1a: “Validation at the end” (w/ header check on transit)

- **Encapsulating node:** $ICV = GMAC(Header || IOAM-Data-Fields)$
- **Transit node:**
  - Check header
  - $ICV = GMAC(ICV || IOAM-Data-Fields)$
- **Decapsulating node:**
  - Check the full chain

* Header = only immutable fields/flags

draft-ietf-ippm-ioam-data-integrity (-08):

(+) Header check on transit nodes (although expensive and optional)

(−) A transit node checks the header by recomputing all ICVs up to itself

(−) Each transit node requires the keys from all prior IOAM nodes (no Zero Trust)
Option 1b: “Validation at the end” (w/ header check on transit)

(+ ) One-step header check for transit nodes
(–) Extra Nonce, ICV (space constraints)
(–) The encapsulating node performs 2 GMACs
(–) Each transit node requires the key from the encapsulating node (no Zero Trust)
Do we really need the header check on transit ?!

For IOAM processing: **YES**
For the Integrity Protection of IOAM data: **NO**

… which is fine: the main objective is to protect the integrity of the **data** (not necessarily the header).

Common problem for 1a and 1b with the header check on transit nodes:

- All IOAM nodes receive the key of the encapsulating node
- Need to trust all IOAM nodes (i.e., **will never be a Zero Trust solution**)

→ Zero Trust implies no header check on transit.
Option 2: “Validation at the end” (no header check on transit)

* Header = only a selection of immutable fields required for the interpretation of IOAM data (not for the processing of IOAM)

(+) Faster processing on transit nodes
(+) Zero Trust: IOAM nodes share their respective keys only with the Validator
(−) No header check on transit nodes
Option 3: Neighbor validation

* Header = all header fields (i.e., entire header)

(+) Header check on transit nodes

(−) Each IOAM node requires the keys from all IOAM nodes (no Zero Trust)
Option 4: IPSec

(+) Does not require defining a new protocol

(−) IPSec tunnels configured between all IOAM nodes that match the physical topology/connectivity (all traffic with IOAM runs across the IPSec tunnels)

(−) Each IOAM node requires the keys from all IOAM nodes (no Zero Trust)

(−) May change the path taken by packets
Summary

<table>
<thead>
<tr>
<th>Option</th>
<th># Icv</th>
<th>Header</th>
<th>Transit Check</th>
<th>Zero Trust</th>
<th># Encap</th>
<th>GMAC</th>
<th># Transit</th>
<th># Decap</th>
</tr>
</thead>
<tbody>
<tr>
<td>1a</td>
<td>1</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>1</td>
<td>p</td>
<td>n-1</td>
<td></td>
</tr>
<tr>
<td>1b</td>
<td>2</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>2</td>
<td>2</td>
<td>n</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>1</td>
<td>1</td>
<td>n-1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

n = the number of IOAM nodes involved (from 1 to n)
p = the IOAM node's position (0 <= p < n)
### Summary

<table>
<thead>
<tr>
<th>Option</th>
<th># Icv</th>
<th>Header</th>
<th>Transit Check</th>
<th>Zero Trust</th>
<th># Encap</th>
<th>GMAC</th>
<th># Transit</th>
<th># Decap</th>
</tr>
</thead>
<tbody>
<tr>
<td>1a</td>
<td>1</td>
<td>(✓)</td>
<td>(✓)</td>
<td>✗</td>
<td>1</td>
<td>p</td>
<td>n-1</td>
<td></td>
</tr>
<tr>
<td>1b</td>
<td>2</td>
<td>(✓)</td>
<td>✓</td>
<td>✗</td>
<td>2</td>
<td>2</td>
<td>n</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>(✓)</td>
<td>✗</td>
<td>✓</td>
<td>1</td>
<td>1</td>
<td>n-1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

n = the number of IOAM nodes involved (from 1 to n)
p = the IOAM node's position (0 ≤ p < n)
### Summary

<table>
<thead>
<tr>
<th>Option</th>
<th># Icv</th>
<th>Header protected</th>
<th>Header transit check</th>
<th>Zero Trust</th>
<th># encap</th>
<th>GMAC # transit</th>
<th># decap</th>
</tr>
</thead>
<tbody>
<tr>
<td>1a</td>
<td>1</td>
<td>(✓)</td>
<td>(✓)</td>
<td>X</td>
<td>1</td>
<td>p</td>
<td>n-1</td>
</tr>
<tr>
<td>1b</td>
<td>2</td>
<td>(✓)</td>
<td>✓</td>
<td>X</td>
<td>2</td>
<td>2</td>
<td>n</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>(✓)</td>
<td>X</td>
<td>✓</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

n = the number of IOAM nodes involved (from 1 to n)
p = the IOAM node’s position (0 <= p < n)
Results

Encapsulating node

Decapsulating node

Transit node
… decapsulating node: Option 2, “n” validations
Conclusion

- draft-ietf-ippm-ioam-data-integrity (-09) now specifies “Option 2” only
  - “Option 1a” abandoned
  - Provides performance risk mitigation for the decapsulating node (i.e., != Validator)
- “Option 3” can be defined later in a separate document
- “Option 1b” and “Option 4” not worth it… “Option 3” is equivalent
  - “Option 4” could be useful for an Inter-Domain use case (secure data transfer from A to B)
- Overall: a story of compromise (no perfect solution)