ESP Header Compression (EHC)

draft-mglt-ipsecme-diet-esp-07
Tobias Guggemos
ipsecme@IETF 104
28.03.2019
Can you imagine a scenario in IoT, where IPsec/ESP would be useful?
Scenarios for IPsec

- Communication with a server (data center or controller)
- Device to Device (D2D) communication
- Inter-Domain communication (Smart Fabrics/Cities/etc.)
- Multicast / Group Communication
- Long term sessionless communications

Tobias Guggemos
IPsec’s Advantage: Flexibility of Key Exchange

• Minimal IKEv2 is already there

• G-IKEv2 works quite well, too

• HIP is standardized for IEEE 802.15.4

• static SAs
So why is IPsec/ESP not widely deployed in IoT?
Problem: Packet Size

Energy cost of IoT communication:
• Increases with the number of radio frames
• Full / empty radio frames have the same cost
• Security protocol overhead may be larger than a radio frame
• Typical payloads:
  • 802.15.4: 102 Byte
  • LORA: 59 – 230 Byte
  • SigFox: 12 Byte
  • Bluetooth LE: >23 Byte
Solution: Header Compression (1)

Stateless (6LoWPAN)

- Compression Information is sent along with the packet on the wire
Solution: Header Compression (2)

Statefull: RObust HC (ROHC), Static Context HC (SCHC)
• Compression is agreed using a separate channel
• This can be static (SCHC) or dynamic (ROHC)

Works with functions and context:

• **Functions:** compression technique for a specific field:
  • e.g. “rule 1: delete upd destination port if it is equal to 80”

• **Context:** pre-known knowledge for decompression
  • e.g. “if packet is compressed with rule 1, set the udp destination port to 80”
Problems (with IPsec)

Example of an IP Tunnel

- All shown techniques compress between L2 and L3
- At that time, ESP payload is already encrypted

⇒ Only ESP header can be compressed without touching the ESP implementation
Good News: ESP Header Compression (EHC)

😊 IPsec already has a separate channel to agree on (and update) a state
  • IKEv2, G-IKEv2, (even HIP could be used)
  • Static

😊 IPsec already has a **static** state (IPsec SAs)

😊 The state already holds some context (Traffic Selector)

😊 We have done this before (ROHCoverIPsec RFC5856)

• We just need to define how to make use of it!
• (Unfortunately) this requires a few specifications

➤ ESP Header Compression (EHC)
  draft-mglt-ipsecme-diet-esp-07
ESP compression Layers

**Compression**

- ESP payload (IP, UDP, TCP, ...)
- ESP Trailer
- Encryption Information (e.g. IV)
  - Currently not part of EHC
  - Refer to Implicit IV
- ESP Header
  - ICV (currently not part of EHC)

**Tobias Guggemos**

ESP Header Compression (EHC)
EHC Actions

<table>
<thead>
<tr>
<th>Function</th>
<th>Compression</th>
<th>Decompression</th>
</tr>
</thead>
<tbody>
<tr>
<td>send-value</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>elided</td>
<td>Not send</td>
<td>Get from EHC Context</td>
</tr>
<tr>
<td>lsb(_lsb_size)</td>
<td>Sent LSB</td>
<td>Get from EHC Context</td>
</tr>
<tr>
<td>lower</td>
<td>Not send</td>
<td>Get from lower layer</td>
</tr>
<tr>
<td>checksum</td>
<td>Not send</td>
<td>Compute checksum.</td>
</tr>
<tr>
<td>padding(_align)</td>
<td>Compute padding</td>
<td>Get padding</td>
</tr>
</tbody>
</table>

• Define the function, how to compress any field
• Derived from ROHC and SCHC specifications

Some examples:
• send-value: IPv4/TCP Options
• elided: IP address, ports
• lsb: ESP/TCP sequence number
• lower: IP/UDP/TCP length
• Checksum: UDP/TCP checksum
• padding: ESP/TCP padding
## EHC Rules (example Inner IPv6)

### EHC Rules

<table>
<thead>
<tr>
<th>EHC Rule</th>
<th>Field</th>
<th>Action</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP6_OUTER</td>
<td>Version</td>
<td>elided</td>
<td>ip_version</td>
</tr>
<tr>
<td></td>
<td>Traffic Class</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Flow Label</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IP6_VALUE</td>
<td>Version</td>
<td>elided</td>
<td>ip_version</td>
</tr>
<tr>
<td></td>
<td>Traffic Class</td>
<td>elided</td>
<td>ip6_tc</td>
</tr>
<tr>
<td></td>
<td>Flow Label</td>
<td>elided</td>
<td>ip6_fl</td>
</tr>
<tr>
<td>IP6_LENGTH</td>
<td>Payload Length</td>
<td>lower</td>
<td></td>
</tr>
<tr>
<td>IP6_NH</td>
<td>Next Header</td>
<td>elided</td>
<td>l4_proto</td>
</tr>
<tr>
<td>IP6_HL_OUTER</td>
<td>Hop Limit</td>
<td>lower</td>
<td></td>
</tr>
<tr>
<td>IP6_HL_VALUE</td>
<td>Hop Limit</td>
<td>elided</td>
<td>ip6_hl</td>
</tr>
<tr>
<td>IP6_SRC</td>
<td>Source Address</td>
<td>elided</td>
<td>ip6_src</td>
</tr>
<tr>
<td>IP6_DST</td>
<td>Dest. Address</td>
<td>elided</td>
<td>ip6_dst</td>
</tr>
</tbody>
</table>

- Map Actions with the available EHC Context
- Defined for every header we consider compressible
  - ESP
  - IPv6 / IPv4
  - UDP / UDP-Lite
  - TCP
  - Anything else?
EHC Context (example: Inner IPv6)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>In SA</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>ip6_tcfl_comp</td>
<td>No</td>
<td>&quot;Outer&quot;, &quot;Value&quot;, &quot;UnComp&quot;</td>
</tr>
<tr>
<td>ip6_tc</td>
<td>No</td>
<td>IPv6 Traffic Class</td>
</tr>
<tr>
<td>ip6_f1</td>
<td>No</td>
<td>IPv6 Flow Label</td>
</tr>
<tr>
<td>ip6_hl_comp</td>
<td>No</td>
<td>&quot;Outer&quot;, &quot;Value&quot;, &quot;UnComp&quot;</td>
</tr>
<tr>
<td>ip6_hl</td>
<td>No</td>
<td>Hop Limit Value</td>
</tr>
<tr>
<td>ip6_src</td>
<td>Yes</td>
<td>IPv6 Source Address</td>
</tr>
<tr>
<td>ip6_dst</td>
<td>Yes</td>
<td>IPv6 Destination Address</td>
</tr>
</tbody>
</table>

- Defined for every header we consider compressible:
  - ESP
  - IPv6 / IPv4
  - UDP / UDP-Lite
  - TCP
  - Anything else?
EHC Strategy

Problem:
• We still need to exchange (and potentially update) all the context and rules via IKEv2
  • For ESP/IPv6/IPv4/UDP/TCP/UDP-Lite, that’s 44 header fields (and thus rules/context)
• More difficult than necessary, as most the valuable values are already there

Solution:
• Strategy defines, how to pre-fill context and rules from available SA values (e.g. IP addresses in Traffic Selector)
• Currently one Strategy for IoT (Diet-ESP) defined, but easy to extend
  ➔ Exchanges only max. 9 fields (instead of 44!) to build a compression context