IPv6 Extension Header – Processing Implications

That is a known bug in 5.00550. Either an upgrade or a downgrade will fix it.
– Larry Wall

Hagen Paul Pfeifer
hagen.pfeifer@protocollabs.de

ProtocolLabs
http://www.protocollabs.com
Munich, Germany

4. April 2010

Processing

- Network stacks iterate over IPv6 extension headers to obtain
  - the wanted transport protocol (`TCP`, `UDP`, `DCCP`, ...) or
  - in the case of some routers a special IPv6 extension header
- First notable characteristic: difference between protocol and extension header
- Protocol stacks are required to iterate over all extension headers until the required chunk is reached
- Well known extension headers: `IPPROTO_HOPOPTS`, `IPPROTO_ROUTING`, `IPPROTO_FRAGMENT`, `IPPROTO_AH`, `IPPROTO_NONE`, `IPPROTO_DSTOPTS`

Extension Header Structure

- Generic extension header:
  - `uint8_t ip6e_nxt` – next header
  - `uint8_t ip6e_len` – length in units of 8 octets
- Standardization shortcoming:
  - IPSec AH header use another `TLV` length field encoding - beware of this bug!
  - No bug: option length calculated by `len + 1 * 8`, note the 1!
  - Sidenote: fragmentation extension header must be handled in a special way

Problematic Case

- Second notable characteristic: extension headers encode always a next header TLV – Protocols like TCP, UDP, DCCP, . . . do not!
- Problem: you cannot iterate over unknown extension headers – which can be unknown protocols – because you don't know if the current extension header follows the TLV encoding.
- Example: `IPv6` → `Routing Extension Header` → `TCP`
  - IPv6 encode in the next field of the IPv6 header that the Routing Extension Header follows
  - Routing Extension Header encodes that TCP follows
- Problem: if the protocol stacks do not know TCP he wrongly assume that TCP is also a kind of extension header and parses the TLV fields. In the case of TCP the TLV fields (next and size) is maps to the 16 bit source port! The instance will interpret the source port as the next and size – buzzer!

Solution

- Academic
  - In a perfect world the designers should have changed TCP, UDP, DCCP, . . . too! TCP 2.0 should follow the TLV encoding.
- Practical
  - But it is hard to change a existing protocol and furthermore the TLV encoding required at least two additional bytes
  - Solution: differentiate between extension header and protocol. And: protocol stacks must be aware of this differentiation and shall know the protocol in advance.

Shortcoming

- If a unknown extension header is encoded in the packet the stack shall abort the processing and generate a ICMP packet to signal the sender that a particular next header is not understandable.
- This makes it difficult to employ a new extension header apart of the IETF, because vendors will not takes notice of the extension.
- On the other hand: it `was` also possible to take the opposite handling: process all unknown extension header until a well known protocol is found. This makes it possible to use new and unknown extension header, but prevents the deployment of new protocols.
- Another hazzle: a unknown extension header should generate a ICMPv6 packet. But this requires that a stack has a list of known extension headers and known protocols!