A Bound End-to-End Tunnel (BEET) mode

draft-nikander-esp-beet-mode-05

Pekka Nikander and Jan Melén

Presentation outline

- Introduction
- Motivation
- BEET in a nutshell
- Packet formats
- IP header processing
- Example SPD and SA
- Implementations
- Some words about the bigger picture

Introduction

- Augments the ESP tunnel and transport modes
  - Aimed for end-to-end tunnels
  - Limited tunnel mode semantics without the tunnelling overhead
  - Inner IP addresses, seen by the applications, and outer IP addresses, used on the wire, made distinct from each other

Motivation

- Intended to support new uses of ESP
  - Tunnel mode with fixed inner addresses
    - Less overhead and slightly better security
  - Transport mode with external address changes
    - NATS, mobility, multi-homing, etc
    - May require connection latching
  - Identifier / locator split in ESP
    - HIP

Motivation — Save some bytes

- “This is useless, just use tunnel mode!”
- Counter-argument: sometimes bytes matter

Headers	Uncompressed	ROCH
Baseline: IPv4 + TCP	20 + 20	2
IPv4 + ESP + IPv4 + TCP	80	58
IPv4 + ESP + TCP	60	38
IPv6 + ESP + IPv6 + TCP	120	78
IPv6 + ESP + TCP	80	38

51% saving

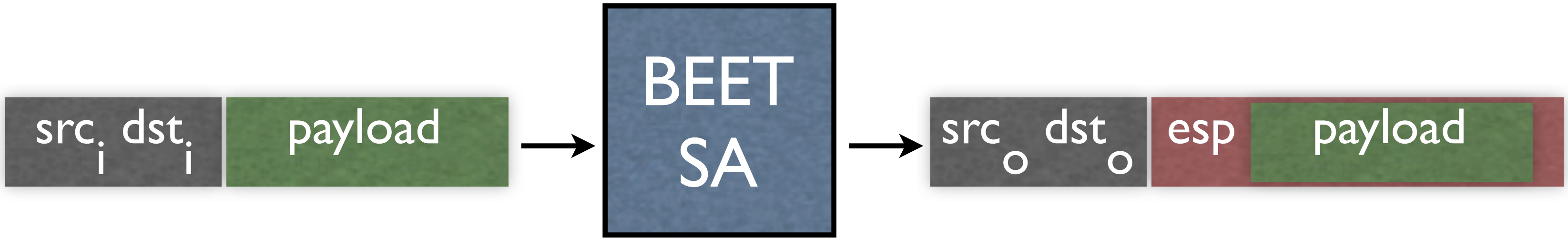
Motivation — Identifier / locator split

- Inner addresses work as end-point identifiers
  - Visible to upper layer protocols
  - No change with mobility / multi-addressing
- Outer addresses work as locators
  - Bound to the topological location
  - Change with mobility / multi-addressing
- Difference to tunnel mode is architectural
  - Inner addresses internal, not visible on wire

BEET in a nutshell

- Transport header but limited tunnel semantics
  - A fixed pair of inner addresses
    - Address ranges not allowed

≈ Transport mode + Bellovin's hostNAT



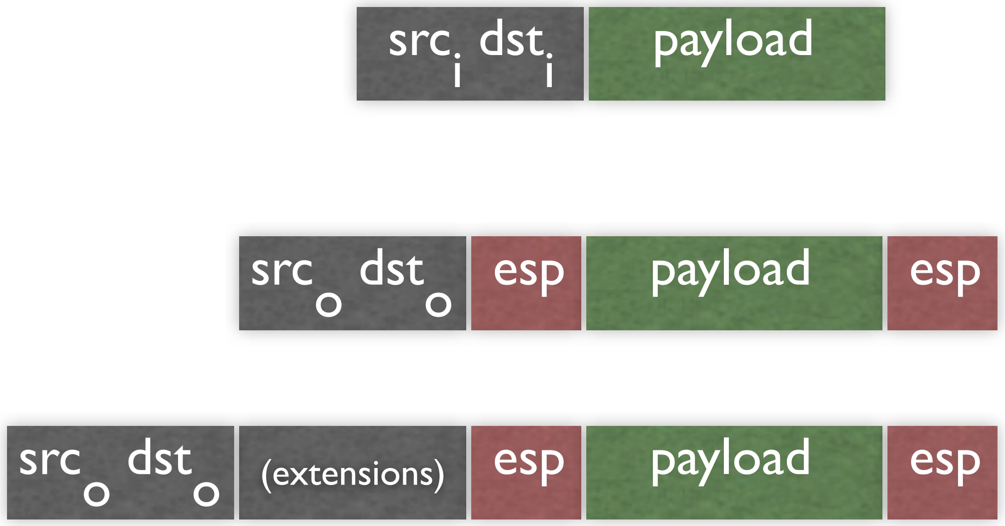
What BEET is not

- Must not be used for non-end-to-end traffic
  - Lack of security analysis, no technical reasons
  - Could be fixed by proper analysis
- Does not obsolete transport or tunnel modes
  - Does not support full tunnel semantics
    - Inner IP addresses strictly bound
    - Only one pair of inner addresses per one BEET SA

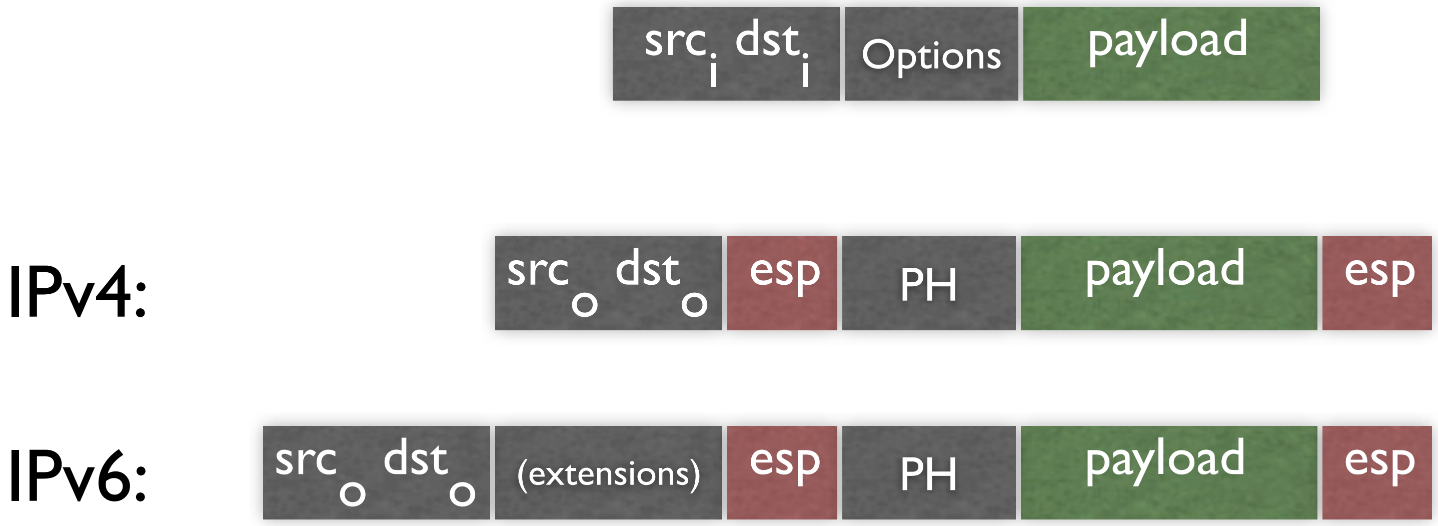
BEET packet format — IPv4 inner addresses, no options

IPv4:

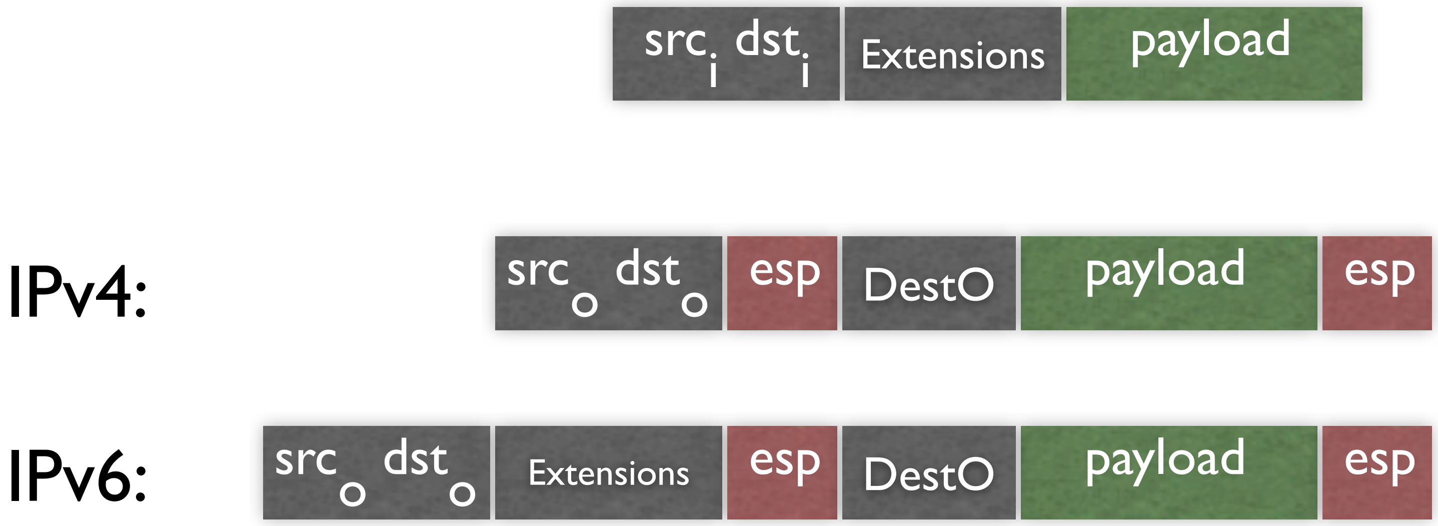
IPv6:



BEET packet format — IPv4 inner addresses, options



BEET packet format — IPv6 inner addresses



IP header processing

- Regular transport mode:
  - IP header is kept intact
- Regular tunnel mode:
  - Outer IP header is created / discarded
- BEET mode:
  - IP header is replaced with another one

Example of BEET policy and SA

```
1188:dd13:8bee:857e:2b90:d10d:7dbd:dad3 [any]
11b3:a8b3:a0fd:df17:8d79:2e48:1733:be53 [any] any out ipsec
        esp/beet/2001:14b8:400:101::50-2001:14b8:400:110::133/require
        created: Mar 15 10:42:05 2006  lastused: Mar 15 10:42:10 2006
        lifetime: 0(s) validtime: 0(s)
        spid=16526 seq=0 pid=70200        refcnt=1

2001:14b8:400:101::50 2001:14b8:400:110::133
        idents: 1188:dd13:8bee:857e:2b90:d10d:7dbd:dad3
        identd: 11b3:a8b3:a0fd:df17:8d79:2e48:1733:be53
        esp mode=beet spi=4133774170(0xf664635a) reqid=0(0x00000000)
        E: rijndael-cbc  f3b4696e 03511cbb 6d20f295 41ea7a25
        A: hmac-sha1  8d932821 d816a293 784a8518 64f31dc9 fb1b0bdf
        seq=0x00000005 replay=0 flags=0x00000000 state=mature
        created: Mar 15 10:42:05 2006   current: Mar 15 10:46:34 2006
        diff: 269(s)    hard: 0(s)      soft: 0(s)
        last: Mar 15 10:42:10 2006      hard: 0(s)      soft: 0(s)
        current: 540(bytes)     hard: 0(bytes)  soft: 0(bytes)
        allocated: 5    hard: 0 soft: 0
        sadb_seq=1 pid=70217 refcnt=2
```

Implementations

- Two independent, interoperating implementations
  - In both cases HIP as the primary motivation

- HIP4inter.net, runs on FreeBSD
  - http://www.hip4inter.net
- HIPL, runs on Linux
  - http://hipl.hiit.fi/hipl/

Summary

- New mode for ESP (and perhaps AH, too)
  - Tunnel semantics, inner and outer addresses
    - Fixed inner addresses, no address ranges
  - Transport mode header structure
- Up to 51% header savings
- Easier dealing with NATs, mobility, multi-homing
- Facilitates identifier / locator separation
- Minimal added complexity: ~100 lines of code