        A Bound End-to-End Tunnel (BEET) mode for ESP
                draft-nikander-esp-beet-mode-04

Status of this Memo

Copyright Notice

Abstract

   This document specifies a new mode, called Bound End-to-End Tunnel
   (BEET) mode, for IPsec ESP. The new mode augments the existing ESP
   tunnel and transport modes. For end-to-end tunnels, the new mode
   provides limited tunnel mode semantics without the regular tunnel
   mode overhead. The mode is intended to support new uses of ESP,
   including mobility and multi-address multi-homing.

Table of Contents

[1](#). Introduction

The current IPsec ESP specification [5] defines two modes of
operation: tunnel mode and transport mode. The tunnel mode is mainly
intended for non-end-to-end use where one or both of the ends of the
ESP Security Associations (SAs) are located in security gateways,
separate from the actual end-nodes. The transport mode is intended
for end-to-end use, where both ends of the security association are
terminated at the end-nodes themselves.

This document defines a new mode for ESP, called Bound End-to-End
Tunnel (BEET) mode. The purpose of the mode is to provide limited
tunnel mode semantics without the overhead associated with the
regular tunnel mode. As the name states, the BEET mode is intended
solely for end-to-end use. It provides tunnel mode semantics in the
sense that the IP addresses seen by the applications and the IP
addresses used on the wire are distinct from each other, providing
the illusion that the application level IP addresses are tunneled
over the network level IP addresses. However, the mode does not
support full tunnel semantics. More specifically, the IP addresses
as seen by the application are strictly bound, and only one pair of
bound inner addresses can be used on any given BEET mode Security
Association. This is in contrast to the regular tunnel mode, where
the inner IP addresses can be any addresses from a defined range.

A BEET mode Security Associations records two pairs of IP addresses,
called inner addresses and outer addresses. The inner addresses are
what the applications see. The outer addresses are what appear on
the wire. Since the inner addresses are fixed for the lifetime of
the Security Association, they need not to be sent in individual
packets. Instead, they are set up as the Security Associations are
created, they are verified when packets are sent, and they are
restored as packets are received.

This all gives the BEET mode the efficiency of transport mode with a
limited set of end-to-end tunnel semantics. The efficiency is
accomplished by removing the inner IP header from the packet that is
transported on the wire. Due to removal of inner IP header, the TTL
of tunneled packet is reduced by every router on the path as the TTL
value is copied from inner to outer header by the sender and vice
versa by the receiver. The semantics of BEET mode is limited in the
sense that only one fixed pair of inner addresses are allowed. The
outer addresses may change over the life time of the SA, but the
inner addresses cannot. If a new pair of inner addresses is needed,
a new pair of BEET mode Security Associations must be established, or
the regular tunnel mode must be used. However, in the cases
considered, a single pair of security associations is usually
sufficient between any single pair of nodes.

[2](#). Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119](#) [[2](#)].

This document contains both normative and informative sections. The normative sections define the BEET mode. The informative sections provide background information that aim to motivate the need for the new mode. Whenever it may not be clear from the context whether a given major section is normative or informative, it is defined in the beginning of the section.

[2.1](#). Terminology

In this section we define the terms specific to this document. This section is normative.

Inner IP address
   An IP address as seen by applications, stored in TCB or other upper layer data structures, and processed by the IP stack prior to ESP processing in the output side and after ESP processing in the input side.

Outer IP address
   An IP address seen in the wire and processed by the IP stack after ESP processing in the output side and before ESP processing in the input side.

Inner IP header
   An IP header that contains inner IP addresses. In some cases an inner IP header may be represented as an internal data structure containing the data equivalent to an IP header.

Outer IP header
   An IP header that contains outer IP addresses. In some cases an outer IP header may be represented as an internal data structure containing the data equivalent to an IP header.

[3](#). **Background**

   For a number of years people have been talking about using IPsec for
   other purposes than VPN. In fact, the current specifications do
   provide support for end-to-end protection of data. However, that
   mode is rarely used, for a number of reasons [6], [7]. One of the
   reasons, though, seems to be address agility. That is, due to NAT,
   mobility, multi-address multi-homing, etc., the addresses that are
   used actually on the wire do not necessarily match with the addresses
   that the applications expect to see. In the NAT case the addresses
   are changed on the fly, thereby invalidating any transport mode
   checksums (unless, of course, a tunnel is used). Mobile nodes change
   their addresses periodically, and the existing applications rarely
   survive the address changes without some help, e.g., Mobile IP.
   Multi-addressing based multi-homed nodes would prefer to keep their
   connections active even when the primary (or currently used) IP
   address becomes unusable in the face of an network outage.

   Based on the reasons above, there is clearly a need for a mode of
   communication where the addresses that the applications see are
   distinct from the addresses that are actually used in the wire. The
   current IPsec tunnel mode provides the required functionality, but at
   the cost of additional overhead in terms of larger packets and more
   complicated processing.

[3.1](#). **Related work**

   The basic idea captured by this draft has been floating around for a
   long time. Steven Bellovin's HostNAT talk [8] at the Los Angeles
   IETF is an early example. After that, basically the same idea has
   surfaced several times. Perhaps the most concrete current proposal
   is the Host Identity Protocol (HIP) [11], where BEET mode ESP
   processing is an integral part of the overall protocol.

[4](#). **Use scenarios**

   In this section we describe a number of possible use scenarios. None
   of these use scenarios are meant to be complete specifications on how
   exactly to support the functionality. Separate specifications are
   needed for that. Instead, the purpose of this section is to discuss
   the overall benefits of the BEET mode, and to lay out a road map for
   possible future documents. This section is informative.

[4.1](#). **NAT traversal**

   NAT traversal is currently a major problem in IPsec. It is not
   sufficient to encapsulate the packets into UDP; additionally, tunnel
   mode must be used. Tunnel mode is required since the outer IP
   addresses at the ends of the protected connection differ. If
   transport mode was used, the differing IP addresses would lead to
   failing upper layer TCP/UDP checksums.

   The BEET mode provides sufficient tunnel mode semantics without the
   packet overhead of the tunnel mode. A pair of BEET mode SAs can be
   effectively used to "un-NAT" packets that have been NATed during
   their travel through the network. Figure 1 illustrates the process.

```
                            Packet contents on a client -> server packet
     +--------+
     | Client |             src = 131.160.175.2 dst = 129.15.6.1 clear text
     +--------+          ^
         | 10.0.0.1      |
         |               |  src = 10.0.0.1      dst = 129.15.6.1 ESP
         |               |
      +-----+            |
      | NAT |           SAs
      +-----+            |
         | 131.160.175.2 |
         |               |  src = 131.160.175.2 dst = 129.15.6.1 ESP
         | 129.15.5.1    |
     +--------+          v
     | Server |             src = 131.160.175.2 dst = 129.15.6.1 clear text
     +--------+ address from unicast SA lookup
```

   Figure 1

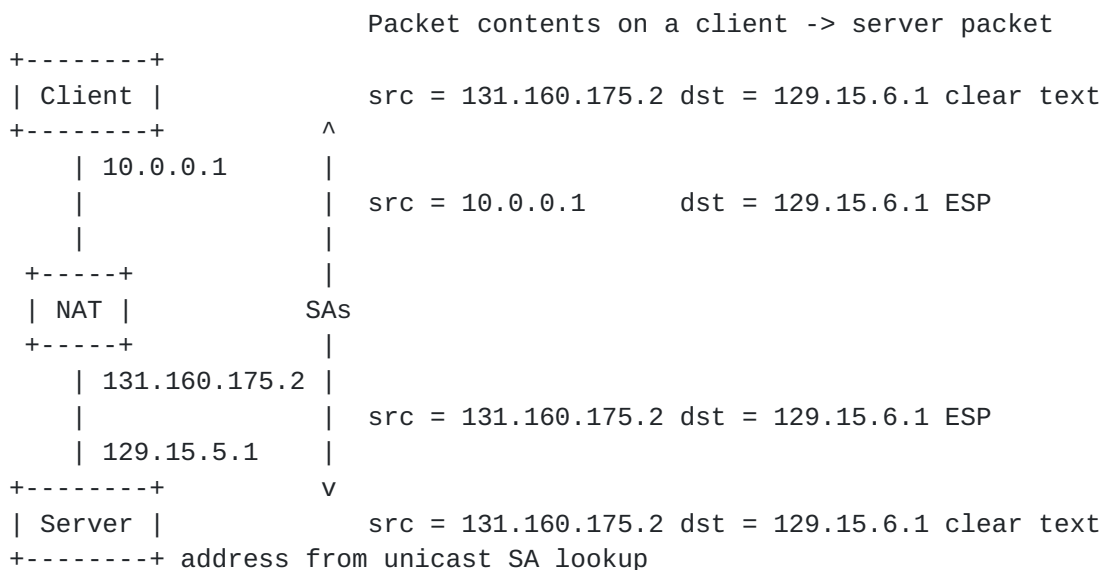   A drawback in this scheme is that the Client must either know its
   public IP address, or it must rely on the Server to tell what address
   to use. It must be noticed that if the NAT box is mapping several
   internal IP addresses into a single public address, the public
   address cannot be directly used. In that case the client and server
   need to agree on a unique address, to be used to internally represent

the client. It must be pointed out that such an address is
semantically very similar to a Mobile IP home address. The details
of such address agreement are beyond the scope of this document.

4.2. Mobile IP

In Mobile IP, the BEET mode could be used instead of the currently
defined wire formats. If the hosts would be using end-to-end ESP
anyway, this has the benefit of saving the space that would otherwise
be taken by the standard Mobile IP wire formats. Furthermore, in
BEET the inner IP header does not actually appear in the wire format.
Effectively, this makes BEET as space efficient for mobile nodes as
the standard ESP transport mode is today between fixed hosts.

Instead of having a separate Binding Cache, the nodes could include
the address translation information into a pair of BEET mode security
associations.

4.2.1. Mobile IPv4

In the current Mobile IPv4, two different wire formats are used,
depending on whether there is a NAT device between the communicating
hosts or not. See Figure 2, below.

Mobile IPv4 wire format without NAT traversal

   IP(CoA->HA) | IP(HoA->CN) | payload

Mobile IPv4 wire format with NAT traversal

   IP(CoA->HA) | UDP(any->434) | MIP header | IP(HoA->CN) | payload

(where the MIP header is a minimal 4 octet header)

     [Figure courtesy to Sami Vaarala.]

Figure 2

It is required that the inner address representing the mobile node,
as seen by the application, is always the home address. That is,
from the application point of view, the packets flow between the home
address and the correspondent node address.

If IPsec is used to protect the traffic between the Mobile Node and
the Correspondent node, ESP transport mode can be used. However, the
transport mode ESP packet is enclosed into an IP-over-IP wrapper at
the home agent, see Figure 3.

Current Mobile IPv4 wire format with end-to-end ESP transport mode:

```
  CN -> HA:                  IP(CN->HoA) | ESP | payload | ESP trailer
  HA -> MN: IP(HA->CoA) | IP(CN->HoA) | ESP | payload | ESP trailer

  MN -> HA: IP(CoA->HA) | IP(HoA->CN) | ESP | payload | ESP trailer
  HA -> CN:                  IP(HoA->CN) | ESP | payload | ESP trailer
```

Proposed Mobile IPv4 wire format with ESP BEET mode:

```
  CN -> HA:                  IP(CN->HoA) | ESP | payload | ESP trailer
  HA -> MN: IP(HA->CoA)              | ESP | payload | ESP trailer

  MN -> HA: IP(CoA->HA)              | ESP | payload | ESP trailer
  HA -> CN:                  IP(HoA->CN) | ESP | payload | ESP trailer
```

Figure 3

In this scenario, the correspondent node does not need to be aware
that the security association is in fact using the BEET mode. If the
home agent and the mobile node co-operate, and the mobile node
implements the BEET semantics, the change could be implemented
transparently to the correspondent node.

The SPI towards the MN MUST be selected so that the HA can
differentiate MNs from each other if they are communicating towards
the same CN. As the SA is anyway end-to-end the HA MAY check during
the key exchange that the selected SPI will not collide with other
MNs.

As multiple CNs may choose same SPI for receiving data from HA, the
HA must implement SPINAT [9] towards MN. Thus, the SPI used to
receive packets from MN at HA would uniquely identify the real
destination CN. The SPI must be negotiated per CN basis but as it is
assumed that there would be a end-to-end SA anyway the amount of
signaling doesn't need to be increased

It should be noticed that the space savings are even larger in the
NAT traversal situation, as is illustrated in Figure Figure 4, below.

Current Mobile IPv4 NAT-traversal wire format with end-to-end transport ESP:

    CN -> HA: IP(CN->HoA) | ESP | payload | ESP
trailer
    HA -> MN: IP(HA->CoA) | UDP | MIP | IP(CN->HoA) | ESP | payload | ESP
trailer

    MN -> HA: IP(CoA->HA) | UDP | MIP | IP(HoA->CN) | ESP | payload | ESP
trailer
    HA -> CN: IP(HoA->CN) | ESP | payload | ESP
trailer

Proposed Mobile IPv4 NAT-traversal wire format with BEET ESP:

    CN -> HA: IP(CN->HoA) | ESP | payload | ESP trailer
    HA -> MN: IP(HA->CoA) | UDP | ESP | payload | ESP trailer

    MN -> HA: IP(CoA->HA) | UDP | ESP | payload | ESP trailer
    HA -> CN: IP(HoA->CN) | ESP | payload | ESP trailer

Figure 4

In the NAT traversal case the HA doesn't have to implement the SPINAT
because the CN MAY be piggy packed in the UDP source and destination
IP and port information. Two separate UDP connections MAY not have
the same source and destination IP and port pairs thus the UDP
connection will identify the CN uniquely.

4.2.2. Mobile IPv4 route optimization

BEET can be used for route optimization purposes as the outer IP
address can always be set to as the current CoA of the MN. Likewise
the MN can set the outer address as the address of the CN instead of
the HA's address, although this would require that both ends support
BEET mode. Binding updates would be sent to the CN as well instead
of just updating the location on HA. Revealing MN's real location to
the CN might not always be desirable.

4.2.3. Mobile IPv6

Triangular routing in Mobile IPv6 is similar to that of Mobile IPv4.
However, the tunnel between the home agent and the mobile node is an
ESP tunnel instead of being a plain IP-over-IP tunnel. However, if
BEET mode was used between the correspondent node and the mobile

node, the ESP tunnel between the home agent and the mobile node would
not bring any additional protection to the payload data. Thus, in
that case BEET could replace the ESP tunnel, similar to the IPv4

case, illustrated in Figure 3 above.

Mobile IPv6 Route Optimization uses a Type 2 Routing Header (RH) and
Home Address Option (HAO) in the packet wire format. However, it can
be argued that the semantics of these options is equivalent to a
optimized point-to-point tunnel. That is, the Type 2 RH defines the
real destination address of a packet, thereby effectively creating a
partial tunnel where the inner and outer source addresses are
identical but the destination addresses differ. Similarly, the Home
Address Option defines the real source address of the packet, again
creating a partial tunnel. The only difference is that this time the
inner and outer destination addresses are identical but the source
addresses differ.

Thus, for Mobile IPv6, BEET mode would define a different wire format
for the payload packets. Instead of using Type 2 RH and HAO, the
packets could be encapsulated into a BEET mode ESP tunnel. In the
case that ESP is used anyway, this has the advantage that the
standard Mobile IPv6 extra headers are not needed, thereby saving
bytes in the headers. Compared to tunnel mode ESP, BEET mode has the
advantage that the inner IP header is not needed.

In Mobile IPv6, mobility management can be implemented just as
before, using the HoTI/CoTI, HoT/CoT and Binding Update (BU)
messages. The difference would lay in handling Binding Updates. If
BEET mode was used, processing Binding Updates would change the outer
IP addresses in the BEET mode Security Associations instead of
changing the Binding Cache.

4.3. End-node multi-address multi-homing

The BEET mode provides for limited end-node multi-address multi-
homing. It semantically provides a tunnel between the end-hosts,
with fixed inner IP addresses. This allows a multi-homed host to use
different outer IP addresses in different packets, without any notice
by the upper layer protocols. The upper layer protocols see the
inner IP address at all times. Thus, this limited form of multi-
homing has no affect on the applications, which seemingly communicate
over fixed IP addresses all the time.

Implementing this kind of limited multi-homing support would require
a small change to the current IPsec SPD and SA implementations.
Currently the incoming SA selection is based on the SPI and
destination address, with the implicit assumption that there is only
one possible destination address for each incoming SA. In a multi-
homed host it would be desirable to have multiple destination
addresses associated with the SA, thereby allowing the same SA to be
used independent on the actual destination address in the packets.

Removing the destination address from unicast SA lookup is already
being proposed in the current ESP draft [5].

If it is considered undesirable to change the implementations to
support multiple alternative destination addresses, it would still be
possible to support limited multi-homing by creating several parallel
SAs, one for each destination address. Each of these SAs would have
identical inner addresses. Effectively, this would distribute the
tunnel over multiple SAs.

In this latter implementation, the outgoing SA processing becomes
more complex. Selecting the outgoing SA does not depend only on the
inner IP addresses but also on the outer destination address.
Selecting the outer destination address depends on the current multi-
homing situation. This creates a situation where the SA processing
must be deferred after selecting the actual outer address to be used.
This might be difficult in some implementations.

4.4. Host Identity Protocol

The Host Identity Protocol (HIP) is a piece of more recent
development. Its aim is to explore the possibilities created by
separating the end-host identifier and locator of IP addresses.
There are currently five implementations, and the specifications are
being finalized. [10] [11]

In HIP, the TCP and UDP sockets are not bound to IP addresses but to
Host Identifiers (HI). The Host Identifiers create a new independent
name space.

The BEET mode supports HIP by defining the inner tunnel in terms of
Host Identifiers and the outer tunnel in terms of standard IP
addresses. In that way all processing prior to outgoing ESP and
after incoming ESP uses Host Identifiers. The wire format packets
use standard IP addresses and ESP transport packet format.

5. Protocol definition

In this section we define the exact protocol formats and operations.
This section is normative.

5.1. Changes to Security Association data structures

A BEET mode Security Association contains the same data as a regular
tunnel mode Security Association, with the exception that the inner
selectors must be single addresses and cannot be subnets. The data
includes the following:

   A pair of inner IP addresses.

   A pair of outer IP addresses.

   Cryptographic keys and other data as defined in RFC2401 [4]
   Section 4.4.3.

A conforming implementation MAY store the data in a way different
than or similar to a regular tunnel mode Security Association.

Note that in a conforming implementation the inner and outer
addresses MAY belong to different address families. All
implementations that support both IPv4 and IPv6 SHOULD support both
IPv4-over-IPv6 and IPv6-over-IPv4 tunneling.

5.2. Packet format

The wire packet format is identical to the ESP transport mode wire
format as defined in [5] Section 3.1.1. However, the resulting
packet contains outer IP addresses instead of the inner IP addresses
received from the upper layer. The construction of the outer headers
is defined in RFC2401 [4] Section 5.1.2. The following diagram
illustrates ESP BEET mode positioning for typical IPv4 and IPv6
packets.

IPv4 INNER ADDRESSES

      BEFORE APPLYING ESP

 | inner IP hdr | | |
 | | TCP | Data |

      AFTER APPLYING ESP, OUTER v4 ADDRESSES

```
   -------------------------------------------------------
   | outer IP hdr |     |     |        |  ESP    | ESP |
   | (any options) | ESP | TCP | Data | Trailer | ICV |
   -------------------------------------------------------
                        |<---- encryption ---->|
                  |<-------- integrity ------->|
```

       AFTER APPLYING ESP, OUTER v6 ADDRESSES

```
   --------------------------------------------------------
   | outer  | new ext |     |     |        |  ESP   | ESP |
   | IP hdr | hdrs.   | ESP | TCP | Data | Trailer| ICV |
   --------------------------------------------------------
                         |<--- encryption ---->|
                   |<------- integrity ------->|
```

   IPv4 INNER ADDRESSES with options

        BEFORE APPLYING ESP

```
   -------------------------------
   | inner IP hdr |     |      |
   |  + options   | TCP | Data |
   -------------------------------
```

        AFTER APPLYING ESP, OUTER v4 ADDRESSES

```
   --------------------------------------------------------------
   | outer IP hdr |     |     |     |        |  ESP    | ESP |
   | (any options) | ESP | PH  | TCP | Data | Trailer | ICV |
   --------------------------------------------------------------
                        |<------- encryption ------->|
                  |<----------- integrity ---------->|
```

        AFTER APPLYING ESP, OUTER v6 ADDRESSES

```
   ----------------------------------------------------------------
   | outer  | new ext |     |     |     |        |  ESP   | ESP |
   | IP hdr | hdrs.   | ESP | PH  | TCP | Data | Trailer| ICV |
   ----------------------------------------------------------------
                         |<------ encryption ------->|
                   |<---------- integrity --------->|
```

                     PH   Pseudo Header for IPv4 options

   IPv6 INNER ADDRESSES

        BEFORE APPLYING ESP

```
   ---------------------------------------------
   |               | ext hdrs |     |      |
```

```
   | inner IP hdr | if present | TCP | Data |
   --------------------------------------------


      AFTER APPLYING ESP, OUTER v6 ADDRESSES
   ----------------------------------------------------------------
   | outer   | new ext |       | dest |       |       | ESP     | ESP |
   | IP hdr  | hdrs.   | ESP | opts.| TCP | Data | Trailer | ICV |
   ----------------------------------------------------------------
                                    |<---- encryption ---->|
                            |<------- integrity ------>|


      AFTER APPLYING ESP, OUTER v4 ADDRESSES
   -------------------------------------------------------
   | outer   |       | dest |       |       | ESP     | ESP |
   | IP hdr  | ESP | opts.| TCP | Data | Trailer | ICV |
   -------------------------------------------------------
                 |<------- encryption -------->|
           |<----------- integrity ----------->|
```

5.3. Cryptographic processing

   The outgoing packets MUST be protected exactly as in ESP transport
   mode [5]. That is, the upper layer protocol packet is wrapped into
   an ESP header, encrypted, and authenticated exactly as if regular
   transport mode was used. The resulting ESP packet is subject to IP
   header processing as defined in Section 5.4 and Section 5.5. The
   incoming ESP protected messages are verified and decrypted exactly as
   if regular transport mode was used. The resulting clear text packet
   is subject to IP header processing as defined in Section 5.4 and
   Section 5.6.

5.4. IP header processing

   The biggest difference between the BEET mode and the other two modes
   is in IP header processing. In the regular transport mode the IP
   header is kept intact. In the regular tunnel mode an outer IP header
   is created on output and discarded on input. In the BEET mode the IP
   header is replaced with another one on both input and output.

   On the BEET mode output side, the IP header processing MUST first
   ensure that the IP addresses in the original IP header contain the
   inner addresses as specified in the SA. This MAY be ensured by
   proper policy processing, and it is possible that no checks are
   needed at the SA processing time. Once the IP header has been
   verified to contain the right IP inner addresses, it is discarded. A
   new IP header is created, using the discarded inner header as a hint
   for other fields but the IP addresses. The IP addresses in the new
   header MUST be the outer tunnel addresses.

On input side, the received IP header is simply discarded. Since the
packet has been decrypted and verified, no further checks are
necessary. A new IP header, corresponding to a tunnel mode inner
header, is created, using the discarded outer header as a hint for
other fields but the IP addresses. The IP addresses in the new
header MUST be the inner addresses.

As the outer header fields are used as hint for creating inner
header, it must be noted that inner header differs as compared to
tunnel-mode inner header. In BEET mode the inner header will have
the TTL, DF-bit and other option values from the outer header. The
TTL, DF-bit and other option values of the inner header MUST be
processed by the stack.

5.5. Handling of outgoing packets

The outgoing BEET mode packets are processed as follows:

1. The system MUST verify that the IP header contains the inner
    source and destination addresses, exactly as defined in the SA.
    This verification MAY be explicit, or it MAY be implicit, for
    example, as a result of prior policy processing. Note that in
    some implementations there may be no real IP header at this time
    but the source and destination addresses may be carried out-of-
    band. In case the source address is still unassigned, it SHOULD
    be ensured that the designated inner source address would be
    selected at a later stage.

2. The IP payload (the contents of the packet beyond the IP header)
    is wrapped into an ESP header as defined in [5] Section 3.3.

3. A new IP header is constructed, replacing the original one. The
    new IP header MUST contain the outer source and destination
    addresses, as defined in the SA. Note that in some
    implementations there may be no real IP header at this time but
    the source and destination addresses may be carried out-of-band.
    In the case where the source address must be left unassigned, it
    SHOULD be made sure that the right source address is selected at
    a later stage. Other than the addresses, it is RECOMMENDED that
    the new IP header copies the fields from the original IP header.

4. If there are any IPv4 options in the original packet, it is
    RECOMMENDED that they are discarded. If the inner header
    contains one or more options that need to be transported between
    the tunnel end-points, sender MUST encapsulate the options as
    defined in Section 5.7

Instead of literally discarding the IP header and constructing a new

one, a conforming implementation MAY simply replace the addresses in
an existing header. However, if the RECOMMENDED feature of allowing
the inner and outer addresses from different address families is
used, this simple strategy does not work.

5.6. Handling of incoming packets

The incoming BEET mode packets are processed as follows:

1. The system MUST verify and decrypt the incoming packet
    successfully, as defined in [5] section 3.4. If the verification
    or decryption fails, the packet MUST be discarded.

2. The original IP header is simply discarded, without any checks.
    Since the ESP verification succeeded, the packet can be safely
    assumed to have arrived from the right sender.

3. A new IP header is constructed, replacing the original one. The
    new IP header MUST contain the inner source and destination
    addresses, as defined in the SA. If the sender has set the ESP
    next protocol field to 94 and included the pseudo header as
    described in Section 5.7, the receiver MUST include the options
    after the constructed IP header. Note, that in some
    implementations the real IP header may have already been
    discarded and the source and destination addresses are carried
    out-of-band. In such case the out-of-band addresses MUST be the
    inner addresses. Other than the addresses, it is RECOMMENDED
    that the new IP header copies the fields from the original IP
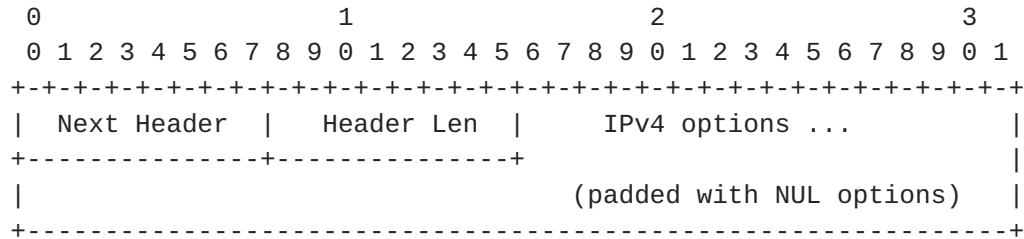   header.

Instead of literally discarding the IP header and constructing a new
one a conforming implementation MAY simply replace the addresses in
an existing header. However, if the RECOMMENDED feature of allowing
the inner and outer addresses from different address families is
used, this simple strategy does not work.

5.7. IPv4 options handling

In BEET mode if IPv4 options are transported inside the tunnel the
sender MUST include a pseudo-header after ESP header. The pseudo-
header identifies that IPv4 options from the original packet are to
be applied on the packet on input side.

The sender MUST set the next protocol field on the ESP header as 94.
The resulting pseudo header including the IPv4 options MUST be padded
to 8 byte boundary. Padding MUST be set to zero which indicates end
of option list as defined in Internet Protocol [1] section 3.1
Internet header format. This way, the receiver is able to identify

where the options ended in the original header. The Header Length
field contains the length of the pseudo header, IPv4 options, and
padding in 8 bytes units.

```
 0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 | Next Header   |  Header Len   |    IPv4 options ...           |
 +---------------+---------------+                               |
 |                                  (padded with NUL options)    |
 +---------------------------------------------------------------+
```

       Next Header   Identifies the data following this header
       Length in bytes   (Header Len + 1) * 8

The receiver MUST remove this pseudo-header as a part of BEET
processing. The IPv4 options included into the pseudo-header MUST be
added after the reconstructed IPv4 (inner) header on the receiving
side.

6. Policy considerations

   In this section we describe how the BEET mode affects on IPsec policy
   processing. This section is normative.

   A BEET Security Association SHOULD NOT be used with NULL
   authentication.

   On the output side, the IPsec policy processing mechanism SHOULD take
   care that only packets with IP addresses matching with the inner
   addresses of a Security Association are passed to that Security
   Association. If the policy mechanism do not provide full assurance
   on this, the SA processing MUST check the addresses. Further policy
   distinction may be specified based on IP version, upper layer
   protocol, and ports. If such restrictions are defined, they MUST be
   enforced.

   On the output side, the policy rules SHOULD prevent any packets
   containing the inner IP addresses pair from escaping to the wire in
   clear text.

   On the input side, there is no policy processing necessary on
   encrypted packets. The SA is found based on the SPI and destination
   address. A single SA MAY be associated with several destination
   addresses. Since the outer IPsec addresses are discarded, and since
   the packet authenticity and integrity is protected by ESP, there is
   no need to check the outer addresses. Since the inner addresses are
   fixed and restored from the SA, there is no need to check them.
   There MAY be further policy rules specifying allowed upper layer
   protocols and ports. If such restrictions are defined, they MUST be
   enforced.

   On the input side, there SHOULD be a policy rule that filters out
   clear text packets that contain the inner addresses.

7. **PF_KEY extensions**

   This section defines the necessary extensions to the PF_KEYv2 API [3]
   to support the BEET mode. This section is informative.

   A BEET mode Security Association is created by specifying the inner
   IP addresses in the PF_KEYv2 Identity extensions, using two new
   identity types. The identity types of the source and destination
   identity extensions MUST be identical, i.e. either IPv4 or IPv6.

```
 #define SADB_X_IDENTTYPE_ADDR          4
 #define SADB_IDENTTYPE_MAX             4
```

   When this new identity type is used, the contents of the identity
   field in the PF_KEY messages MUST be a socket address.

   For SADB_X_IDENTTYPE_ADDR the length of AF_INET type of identity MUST
   be the length of struct sockaddr_in. Thus the length for AF_INET6
   type of identity MUST be the length of struct sockaddr_in6.

   Additionally, a new IPsec mode is defined in ipsec.h, and used in the
   unspecified but commonly used the PF_KEY extension SADB_X_EXT_SA2
   field sadb_x_sa2_mode.

```
 #define IPSEC_MODE_BEET 4
```

   If an SA is specified using SADB_X_EXT_SA2, if the sadb_x_sa2_mode is
   IPSEC_MODE_BEET, and if the source and destination identities are
   defined in terms of SADB_IDENTTYPE_ADDR, then the BEET mode MUST be
   used. If an SA is specified using SADB_X_EXT_SA2, if the
   sadb_x_sa2_mode is IPSEC_MODE_BEET, and if the identities are defined
   in terms (other than the new type defined above) that exactly match
   to single IPv4 or IPv6 addresses, then the BEET mode SHOULD be used.

8. **New requirements on Key Management protocols**

   In this section we discuss the requirements that the new mode places
   upon existing and new key management protocols. This section is
   informative.

   In order to provide support for the BEET mode, key agreement protocol
   implementations must understand the existence of such a mode. In
   some situations it is sufficient that the BEET mode is implemented at
   the IPsec ESP level only at one end as long as the key management is
   aware of its usage. For example, the NAT scenario described in
   Section 4.1 does not require a BEET ESP implementation at the server
   end. It is sufficient that the client implements the BEET mode; in
   fact, if the client somehow knows its public IP address it may be
   able to set up the BEET mode security associations without any
   explicit concent on the server end. On the other hand, if the client
   does not know its public IP address, it needs help from the server in
   order to determine it.

   More generally, one can get benefit from the BEET mode only to the
   extend the key management protocol supports it. If the key
   management protocol is fully aware of mobility and multi-homing
   issues, and provides facilities for signaling changes in the current
   connectivity situation, it is relatively easy to implement end-node
   mobility and multi-address multi-homing with BEET. An example of
   such usage is HIP [11].

9. Implementing the functionality with other means

   It is currently possible to implement the equivalent of BEET mode by
   using transport mode ESP and explicit network address translation at
   the end-hosts themselves. In this section we briefly compare BEET
   mode and transport mode ESP with explicit network address translation
   alternatives. The purpose of this section is to give background
   information for security considerations. This section is
   informative.

   In an implementation using the BEET mode, the input side IP address
   translation is integrated with the decryption and integrity
   verification processing. The packet is passed and given the inner
   addresses if and only if it is correctly decrypted and verified. A
   typical IPsec SPD implementation would prohibit receiving unprotected
   IP packets that use the inner addresses on the wire, as it is done in
   the regular tunnel mode. At the same time, any other uses of the
   outer addresses would be trivial; passing a packet to the SA requires
   both that the packet has an ESP header and that the SPI matches.

   In an implementation based on explicit network address translation
   and transport mode ESP, the address translation and cryptographic
   processing are completely separate. In practise, the host must
   translate the outer IP address into the inner IP addresses before the
   packet is passed to IPsec. (The other way around may not be secure,
   since there would be no way for the address translation process to
   know if the packet was received through IPsec processing or if it was
   received via some other means.) Using the outer addresses for other
   purposes may be hard, depending on the implementation of the address
   translation mechanism. In particular, using the outer addresses on
   other ESP SAs may be hard, since the typical address translation
   mechanisms are only configured on protocol level ESP or not ESP and
   typically do not understand SPIs.

   At the output side, a BEET mode implementation takes care of
   translating the inner addresses to outer addresses, as a part of the
   encryption process. The IPsec SPD contains necessary entries that
   make sure that the inner addresses never leak.

   In an implementation based on transport mode ESP and explicit network
   address translation, the output packets would be passed with inner
   addresses from IPsec to the address translation mechanism. The
   address translation mechanism will then translate the inner addresses
   to outer addresses. While this does not prevent usage of the outer
   addresses for other purposes, the configuration is brittle and error
   prone. If there are mistakes at the IPsec configuration, the address
   translation mechanism may translate unprotected packets, leading to
   potential confusion. If there are mistakes at the address

translation side, the inner addresses may leak to the network.

10. Security Considerations

In this section we discuss the security properties of the BEET mode, discussing some limitations [12]. This section is normative.

There are no known new vulnerabilities that the introduction of the BEET mode would create.

It is currently possible to implement the equivalent of BEET mode by using transport mode ESP and explicit network address translation at the end-hosts themselves. However, such an implementation is more complex, less flexible, and potentially more vulnerable to security problems that are caused by misconfigurations; see Section 9.

The main security benefit is an operational one. To implement the same functionality without the BEET mode typically requires configuring three different, unrelated components in the hosts.

   The transport mode ESP SAs must be configured.

   A host based NAT function must be configured to properly translate between the inner and outer addresses.

   A host firewall must be configured to properly filter out packets so that inner addresses do not leak in or out.

While it may be possible to configure these components to achieve the same functionality, such a configuration is error prone, increasing the probability of security vulnerabilities. An integrated BEET mode implementation is less prone to configuration mistakes. Furthermore, it would be fairly hard to implement portable key management protocols that would be able to configure all of the required components at the same time. On the other hand, it would be easy to provide a portable key management protocol implementation that would be able to configure BEET mode SAs through the specified PF_KEY extensions.

Since the BEET security associations have the semantics of a fixed, point-to-point tunnel between two IP addresses, it is possible to place one or both of the tunnel end points into other nodes but those that actually "possess" the inner IP addresses, i.e., to implement a BEET mode proxy. However, since such usage defeats the security benefits of combined ESP and hostNAT processing, as discussed above, the implementations SHOULD NOT support such usage.

As in the BEET mode the outer header source address is not checked at
the input handling, there is the potential possibility a DoS attack
where the attacker sends random packets that match with the SPI of
some BEET mode SA. This kind of attack would cause the victim to
perform unnecessary integrity checks that would result in a failure.
If this kind of behaviour is detected, the node may request rekeying
from the Key Management Protocol, and after rekeying, if the attacker
was not on the path, the new SPI value would not be known by the
attacker.

11. IANA Considerations

The PF_KEYv2 interface should probably have an IANA registry.

[12](). Acknowledgments

   During the 56th IETF meeting in San Francisco and afterwards, the
   following people made comments on the ideas, helping the author to
   write the draft: Jari Arkko, Steven Bellovin, Charlie Kaufman, Tero
   Kivinen, Cheryl Madson, Andrew McGrecor, Robert Moskowitz, Michael
   Richardson, Timothy Shepard, Jukka Ylitalo, Sami Vaarala, Petri
   Jokela, Herbert Xu, Miika Komu.

   The author ows special thanks to Derek Atkins and Steve Kent, who
   strongly opposed the idea during the San Francisco IETF, and thereby
   forced writing a high quality initial draft.

13. References

13.1. Normative references

[1] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.

[2] Bradner, S., "Key words for use in RFCs to Indicate Requirement
     Levels", BCP 14, RFC 2119, March 1997.

[3] McDonald, D., Metz, C., and B. Phan, "PF_KEY Key Management API,
     Version 2", RFC 2367, July 1998.

[4] Kent, S. and R. Atkinson, "Security Architecture for the
     Internet Protocol", RFC 2401, November 1998.

[5] Kent, S., "IP Encapsulating Security Payload (ESP)",
     draft-ietf-ipsec-esp-v3-05 (work in progress), April 2003.

13.2. Informative references

[6]   Arkko, J. and P. Nikander, "Limitations in IPsec Policy",
      Security Protocols 11th International Workshop, Cambridge, UK,
      April 2-4, 2003, LNCS to be published, Springer, April 2003.

[7]   Ionnadis, J., "Why we still don't have IPsec", Network and
      Distributed Systems Security Symposium (NDSS'03), Internet
      Society, February 2003.

[8]   Bellovin, S., "EIDs, IPsec, and HostNAT", IETF 41th,
      March 1998.

[9]   Ylitalo, J., Melen, J., Nikander, P., and V. Torvinen, "Re-
      thinking Security in IP based Micro-Mobility", 7th Information
      Security Conference (ISC'04) , Palo Alto, September 27-29,
      2004, to be published, Springer, September 2004.

[10] Moskowitz, R., "Host Identity Protocol Architecture",
      draft-ietf-hip-arch-01 (work in progress), December 2004.

[11] Moskowitz, R., "Host Identity Protocol", draft-ietf-hip-base-00
      (work in progress), June 2004.

[12] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on
      Security Considerations", draft-iab-sec-cons-03 (work in
      progress), February 2003.

Appendix A. **Implementation experiences**

   We have implemented the BEET mode to the FreeBSD 5.3 KAME stack. Our
   implementation uses the PF_KEYv2 identity extension, as described in
   Section 7.

   The current implementation is based on four hooks placed at the
   strategical locations at the ESP and ip_output processing. We
   support full IPv4/IPv6 conversions, allowing both IPv4-over-IPv6 and
   IPv6-over-IPv4 tunneling. The number of lines changed in the KAME
   policy processing is 36 lines; these changes were necessary to fully
   support the identity extension, which was partly unimplemented in the
   KAME stack. The hooks themselves take 83 lines, and the protocol
   processing code is 450 lines long. About 90% of the protocol
   processing code was copied and pasted from the IPsec tunnel mode and
   transport mode routines, with minimal changes. About 70% of the code
   is needed to implement v4-over-v6 and v6-over-v4 tunneling. The
   number of actual functional lines for the simple v4-over-v4 and v6-
   over-v6 cases is mere 62 lines. The implementation effort took three
   days from two programmers, including writing simple test cases and
   performing rudimentary testing on the implementation to see that it
   works.

   The current implementation is tailored for experimentation. A more
   proper implementation would implement all of the processing as an
   integral part of the IPsec processing. The current KAME code
   supports only two modes. Once the necessary cleanups, such as
   replacing "if" statements with "switch" statements, we expect the
   extra protocol processing code required by the BEET mode to take less
   than 100 lines.

Appendix B. **Garden beets**

   Commonly known as the garden beet, this firm, round root vegetable
   has leafy green tops, which are also edible and highly nutritious.
   The most common color for beets (called "beetroots" in the British
   Isles) is a garnet red. However, they can range in color from deep
   red to white, the most intriguing being the Chioggia (also called
   "candy cane"), with its concentric rings of red and white. Beets are
   available year-round and should be chosen by their firmness and
   smooth skins.

Authors' Addresses

    Pekka Nikander
    Ericsson Research Nomadic Lab
    JORVAS FIN-02420
    FINLAND

    Phone: +358 9 299 1
    Email: pekka.nikander@nomadiclab.com

    Jan Melen
    Ericsson Research Nomadic Lab
    JORVAS FIN-02420
    FINLAND

    Phone: +358 9 299 1
    Email: jan.melen@nomadiclab.com