Workgroup: IPSECME Working Group
Published: 23 October 2023
Intended Status: Standards Track
Expires: 25 April 2024
Authors: A. Antony S. Klassert
secunet secunet

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

Abstract

This document specifies a new mode for IPsec ESP, known as Bound End-to-End Tunnel (BEET) mode. This mode complements the existing ESP tunnel and transport modes, while enhancing end-to-end IPsec usage. It offers the characteristics of the tunnel mode but without its usual overhead. The BEET mode is designed to accommodate evolving applications of ESP, such as minimalist end-to-end tunnel, mobility and multi-address multi-homing. Additionally, this document proposes a new Notify Message, USE_BEET_MODE, for the Internet Key Exchange Protocol Version 2 (IKEv2) specified in [RFC7296], to facilitate BEET mode Security Association negotiation.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 25 April 2024.

Table of Contents

1. Introduction

The current IPsec ESP specification [RFC4303] defines two modes of operation: the tunnel mode and the 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 IPsec 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 BEET 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 BEET mode does not support full tunnel semantics. More specifically, the IP addresses as seen by the application are strictly bound. In BEET mode, only a pair of addresses is negotiated, and only this pair of strictly bound inner addresses MUST be used within a given BEET mode Security Association. This is in contrast to the regular tunnel mode, where an IP address range, source range and destination range, can be negotiated, and the inner IP addresses can be any pair of IP addresses within the negotiated IP address range. A BEET mode Security Association 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 appears on the wire. Since the inner addresses are fixed for the lifetime of the Security Association, they need not be sent in each packet. 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 BEET mode the efficiency of transport mode with a limited set of end-to-end tunnel semantics. The increase efficiency is accomplished by removing the inner IP header from the packet that is transported on the wire. Due to this removal of the inner IP header, the TTL of a tunneled packet is reduced at 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 is allowed. The outer addresses may change over the lifetime of the SA, but the inner addresses cannot. If a new pair of inner addresses is needed, a new BEET mode Security Association must be established. In the cases considered here, a single pair of security associations between any single pair of nodes is usually sufficient.

1.1. Requirements Language

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 RFC 2119 [RFC2119].

1.2. Terminology

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

- *Inner IP address : An IP address as seen by applications, typically stored in TCP or other upper layer data structures. It is 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 on 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 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.

2. Background

For years, the idea of a minimal IPsec tunnel mode for end-to-end security has sparked discussions and innovation. BEET mode has been in use for over a decade. It hs the potential to improve secure data transmission. Once BEET mode is standardized its potential for enhancing secure communications would increase with interoperablity.

2.1. Related work

The basic idea captured by this draft has been floating around for a long time. Steven Bellovin's HostNAT talk [8] and at the Los Angeles IETF is an early example, and this ID is based on BEET mode ID [I-D.nikander-esp-beet-mode]. The same idea has surfaced several times. Perhaps the most concrete current proposal is the Host Identity Protocol (HIP) [RFC5201] and [RFC5202] where BEET-mode ESP processing is an integral part of the overall protocol without IKE negotiation. The updated version of the Host Identity Protocol (HIPv2) [RFC7401] and [RFC7402] recommend the use of BEET mode.

Minimal ESP [RFC9333] can also benefit from the use BEET mode. It reduces the outter packet size by 20 - 40 bytes per packet, by ommiting inner IPv4/IPv6 header, compared to tunnel mode. However [RFC9333] did not recommend the use of BEET mode, as it is not yet standardized

3. Use scenarios

In this section we describe a number of possible use scenarios. None of these 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 BEET mode, and to lay out a road map for possible future documents. This section is informative.

3.1. Minimal IPsec for end-to-end

Over the years, BEET-mode has been widely adopted as a minimal IPsec tunnel for end-to-end scenarios, effectively reducing data transmission over the wire. It also helps alleviate MTU issues. Additionally, BEET is valuable for low-power devices, as it reduces power consumption [RFC9333] and complexity. In situations where devices or IPsec connections are dedicated to a single application or transport protocol, BEET mode simplifies packet processing and conserves energy, especially for lower-powered devices.

3.2. NAT traversal

NAT traversal is currently a major issue in IPsec. Encapsulating packets into UDP using Transport mode may suffice in some cases, but it becomes inadequate when multiple clients are behind a NAT gateway. In such situations, tunnel mode becomes necessary.

BEET mode offers tunnel mode semantics without the additional packet overhead associated with traditional tunnel mode when operating behind a NAT gateway. A pair of BEET-mode Security Associations (SAs) can effectively "un-NAT" packets originationg from behind a NAT gatewa as they traverse the network. [AA do we need this ?? This process is illustrated in Figure 1.]

[SK: The figure and the comments from the old draft is missing here. Is the scenario where a client behind NAT knows it's public address common? Maybe ask Pablo.]

3.3. End-node multi-address multi-homing

BEET mode enables limited end-node multi-address multi-homing. It establishes a tunnel between end-hosts with fixed inner IP addresses, allowing multi-homed hosts to use different outer IP addresses in various packets without impacting upper-layer protocols. Upper-layer protocols consistently see the inner IP address, ensuring seamless communication over fixed IP addresses.

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 [RFC4303]. [AA verify this part]

4. Protocol Definition

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

4.1. Changes to Security Association data structure

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 [RFC4301] Section 4.4.2.

A conforming implementation MAY store the data in a way 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.

4.2. Packet format

The wire packet format is identical to the ESP transport mode wire format as defined in [RFC4303] 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 [RFC4301] Section 5.1.2. The following diagrams illustrates ESP BEET mode positioning for typical IPv4 and IPv6 packets.

[SK: The figures are not consistent. Let's discuss this in a call. Why is dest opts. in IPv6 encrypted, but other ext hdrs are not?

4.3. Inner IPv4 Datagram

Figure 1: IPv4 INNER DATAGRAM BEFORE APPLYING ESP

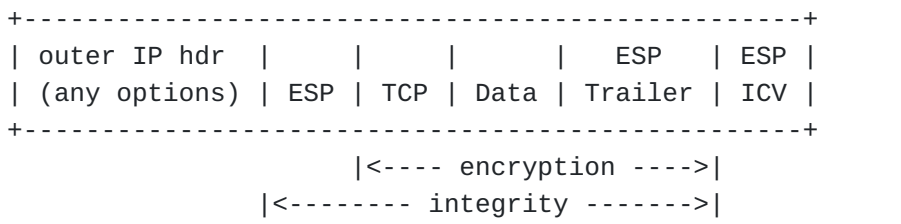

Figure 2: AFTER APPLYING ESP, OUTER v4 ADDRESSES

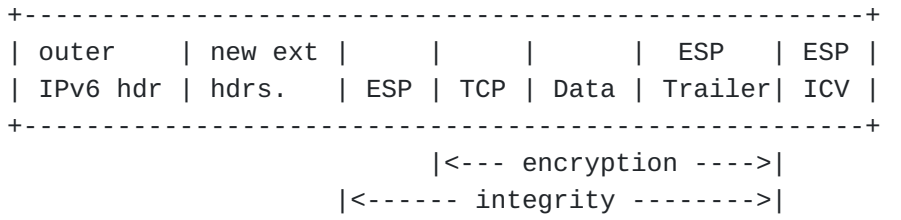

Figure 3: AFTER APPLYING ESP, OUTER v6 ADDRESSES

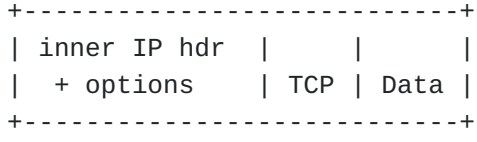

Figure 4: IPv4 INNER DATAGRAM with IP options BEFORE APPLYING ESP

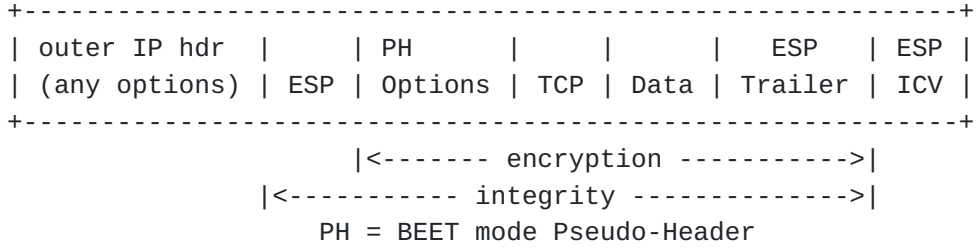

Figure 5: IPv4 AFTER APPLYING ESP, OUTER v4 ADDRESSES INNER IPv4 OPTIONS

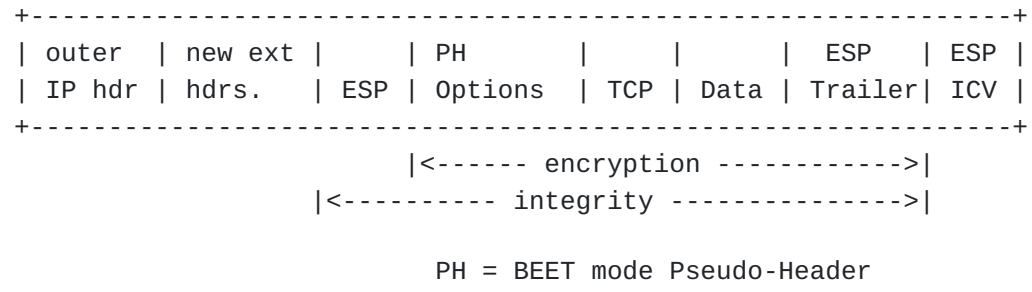

Figure 6: IPv4 + OPTIONS AFTER APPLYING ESP, OUTER IPv6 ADDRESSES

4.4. Inner IPv6 Datagram

Figure 7: IPv6 DATAGRAM BEFORE APPLYING ESP

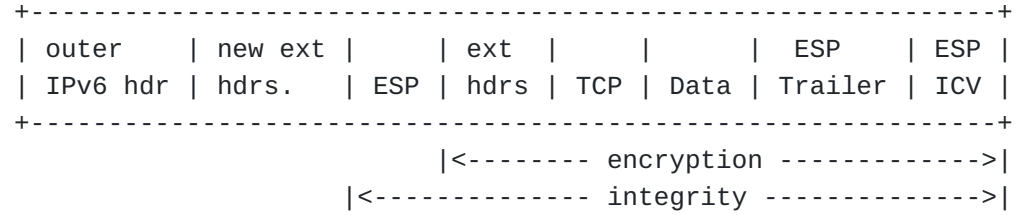

Figure 8: IPv6 DATAGRAM AFTER APPLYING ESP, OUTER IPv6 ADDRESSES

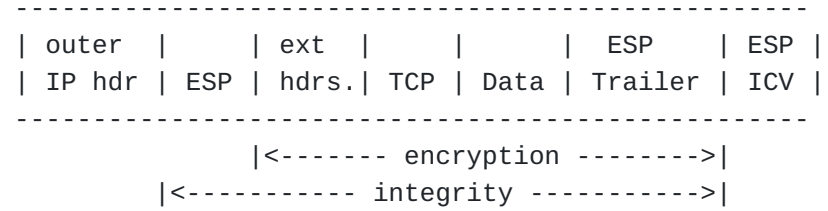

Figure 9: IPv6 DATAGRAM AFTER APPLYING ESP, OUTER IPv4 ADDRESSES

5. Cryptographic processing

The outgoing packets MUST be protected exactly as in ESP transport mode [RFC4303]. 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 6 and Section 7. The incoming ESP protected messages are verified and decrypted exactly as if regular transport mode was used. The resulting cleartext

packet is subject to IP header processing as defined in Section 6 and Section 8

6. IP header processing

The biggest difference between BEET mode and the other two modes lies in the way IP headers are processed. In 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 BEET mode, the IP header is replaced with another one on both input and output.

On the BEET mode output side, the IP 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 derived, using the discarded inner header as a hint for other fields except the IP addresses, the IPv4 options, the IPv6 optional extensions, and the IPv4 fragmentation offset when the packet is a fragment. The IP addresses in the new header MUST be the outer tunnel addresses.

On the 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 derived, using the discarded outer header as a hint for other fields but the IP addresses, the IPv4 options, the IPv6 optional extensions, and the IPv4 fragmentation offset when the packet is a fragment. The IP addresses in the new header MUST be the inner addresses negotiated.

As the outer header fields are used as a hint for creating the inner header, it must be noted that the BEET inner header differs from tunnel mode inner headers. In BEET mode, an inner header will contain 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. [AA would we loose the DF value?? if IPsec gateway has different policy? or is possible to restore the same as inner packet?]

7. Handling of outgoing packets

The outgoing BEET mode packets are processed as follows:

* 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.

*The IP payload (the contents of the packet beyond the IP header) is wrapped into an ESP header as defined in [RFC4303] Section 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 cases where the source address must be left unassigned, it SHOULD be ensured 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, unless the packet is a IP fragment.

*If there are any IPv4 options in the original packet, they are copied into ESP payload. The IPv4 options are tunneled between the tunnel end-points. The sender MUST encapsulate the options as defined in Section 9.

*If the packet is an IPv4 fragment, the flags and the fragment offset field from the inner IP packet MUST NOT be copied. Instead these two fields MUST be encapsulated with a PH header as defined in Section 9.

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.

8. Handling of incoming packets

The incoming BEET mode packets are processed as follows:

1. The system MUST successfully verify and decrypt the incoming packet, as specified in [RFC4303] section 3.4. If the verification or decryption fails, the packet MUST be discarded.

2. The original IP header is discarded without further verification since the ESP verification has already confirmed the packet's integrity and source. 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 9, 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. [AA how about ESP in UDP and mapping changes?]

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

9. IPv4 Pseudo Header

In BEET mode, if IPv4 options or IPv6 optional extensions are transported inside the tunnel, as ESP payload. For IPv4, the sender MUST include a Pseudo Header(PH) after ESP header. The pseudo-headerr indicates the IPv4 packet has options or the fragmentation flags and fragmentation offset is set.

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 octet boundary. The padding length is expressed in octets, valid padding lengths are 0 or 4 octets as the original IPv4 options are already padded to 4 octet boundary. The padding MUST be filled with NOP options as defined in Internet Protocol [RFC791] section 3.1 Internet header format. The padding is added in front of the original options to ensure that the receiver is able to reconstruct the original IPv4 datagram. The Header Length field contains the length of the IPv4 options, and padding in 8 octets units.

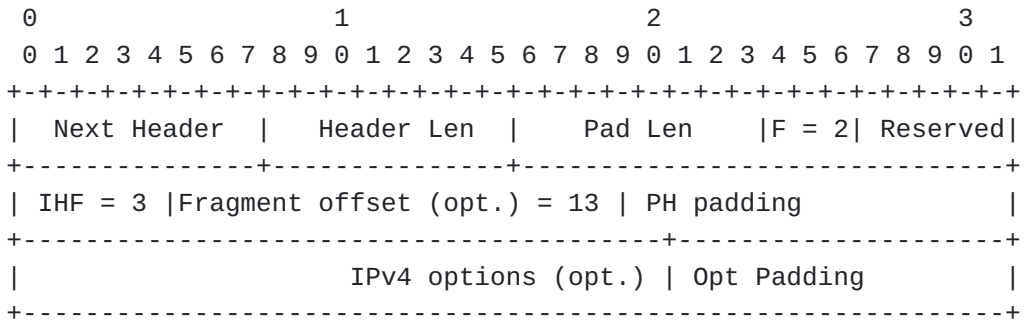

Figure 10: BEET mode pseudo header format

*Next Header - Identifies the data following this header.

*Header Len - 8-bit unsigned integer. Length of the pseudo header in 8-octet units, including IHF and Fragment offset, IPv4 options, when present, not including the first 8 octets.

*Pad Len - 8-bits unsigned integer. Length of padding in octets, valid padding lengths are:

-0, 4 : no IPv4 fragment and options are present [AA Note : old case]

-2, 6 : IPv4 fragment and no IPv4 options.

-2, 6 : IPv4 fragment and options.

*F - 2 bits Flags. 00 IP options, 01 Fragment, 10 IP options and Fragment, 11 Reserved

*Reserved - Lower 6-bits, set to zero

*IHF - Flags from the inner IP header (optional) when F = 01 or 10

*Fragment offset 13 bits - Fragment offset from the inner IP header (optional) when F = 01 or 10

*Padding - 0, 2 or 4 octets of zeros, depending on the fragment flag.

*IPv4 options - IPv4 options from the inner IP header with optional padding. Alligned to 32 bits.

The receiver MUST remove this pseudo-header and padding as a part of BEET processing, in order to reconstruct the original IPv4 datagram. The IPv4 options included in the pseudo-header MUST be added after the reconstructed IPv4 (inner) header on the receiving side. Also copy the flags and Fragment offset when the PH flags is 01, 10.

Note: when the IPv4 options are present, the outer header's IHL would be different from the inner header IHL.

The receiver MUST remove this pseudo-header and padding as a part of BEET processing, in order reconstruct the original IPv4 datagram. The IPv4 options included into the pseudo-header MUST be added after the reconstructed IPv4 (inner) header on the receiving side.

Note: When the pseudo-header is present due to the presence of IPv4 options in the inner IP header, the outer header's IHL (Internet Header Length) would be different from the inner IP header IHL.

10. IPv4 Inner Fragments

When the inner IPv4 datagram is a fragment (as specified by the "more-fragments" flag being set to one [RFC791]), this flag MUST NOT be copied to the outer ESP datagram header. Additionally, for any non-first fragment with a "more-fragments" flag or "fragment offset field," these two fields MUST NOT be copied to the outer IPv4 header of the ESP datagram.

Here are a few possible ways to deal with these IPv4 fragments.

1. Re-assemble the IPv4 fragments, send to ESP, and ESP datagram may be fragmented as required.

2. Drop the IPv4 fragments, i.e., BEET mode IPv4 support for fragments is optional.

3. Copy the fragment flag and offset length from the inner IPv4 header to BEET pseudo-header Section 9.

4. Explore other solutions? [TBD?]

TBD Discuss/Decide which of the above options make sense.

11. IPv6 inner Fragments

It's crucial to highlight that IPv6 uses fragmentation information in a distinct manner than IPv4 [RFC8200] Section 4.5. Specifically, an IPv6 fragment uses IPv6 optional extensions for fragments. BEET mode treats the IPv6 optional extensions as ESP payload and move from inner header to ESP payload.

12. Mixed family IPv4 inside and IPv6 outside

The inner datagram's IP version MUST be independent of the outer IP version. The inner address family and address are taken from the negotiated Traffic Selectors. When the inner datagram contains IPv4 with fragments or IPv4 options, use Section 9.

13. Mixed family IPv6 inside and IPv4 outside

When the inner datagram is an IPv6 datagram with IPv6 optional extensions, copy it into ESP payload Section 11.

14. Policy Considerations

In this section we describe how 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 the inner addresses of a Security Association are passed on to that Security Association. If the policy mechanism does not provide full assurance on this point, 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 pair of inner IP addresses from escaping to the wire in cleartext.

On the input side,no policy processing is necessary for encrypted packets. The SA is deduced from the SPI and destination address. A single SA MAY be associated with several outter destination addresses. Since the outer IPsec addresses are discarded, and since the packet authenticity and integrity are 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 cleartext packets that contain the inner addresses.

15. Security Considerations

In this section we discuss the security properties of the BEET mode, discussing some and point out some of its limitations [RFC3552].

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 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 end0points 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.

Because in BEET mode the outer header source address is not checked at the time of input handling, there is a potential for a DoS attack where the attacker would send random packets that match 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. However, if this kind of behavior is detected, the node may request rekeying using IKEv2 rekey, and after rekeying. If the attacker was not on the path, the new SPI value would not be known by the attacker.

16. IKEv2 Negotiation

When negotiating a Child SA using using IKEv2, the initiator may use the new "USE_BEET_MODE" Notify Message to request a Child SA pair with BEET mode support. The method used is similar to how USE_TRANSPORT_MODE is negotiated, as described in [RFC7296]

To request a BEET-mode SA on the Child SA pair, the initiator MUST include the USE_BEET_MODE Notify Message when requesting a new Child

SA, either during the IKE_AUTH or CREATE_CHILD_SA exchanges. If the request is accepted, the response MUST also include a USE_BEET_MODE notification. If the responder declines and does not include the USE_BEET_MODE notification in the response, the child SA will be established without BEET mode enabled. If this is unacceptable to the initiator, the initiator MUST delete the child SA.

16.1. USE_BEET_MODE Notify Message Payload

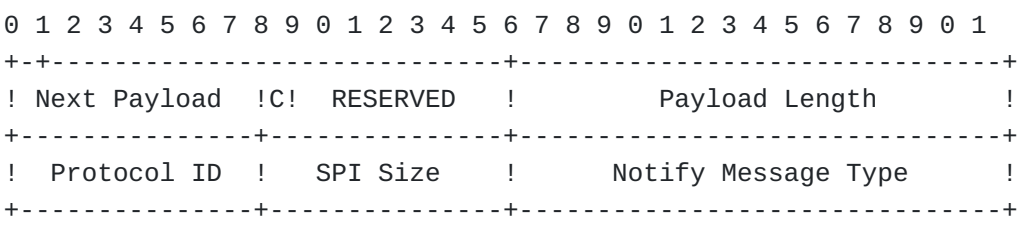

Figure 11

*Payload Length - MUST be 0.

*Protocol ID (1 octet) - MUST be 0. MUST be ignored if not 0.

*SPI Size (1 octet) - MUST be 0. MUST be ignored if not 0.

17. IANA Considerations

This document defines a new IKEv2 Notify Message Type payloads for the IANA "IKEv2 Notify Message Types - Status Types" registry.

Value	Notify Type Messages - Status Types	Reference
[TBD1]	USE_BEET_MODE	[this document]

Figure 12

18. Implementation Status

[Note to RFC Editor: Please remove this section and the reference to [RFC6982] before publication.]

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC7942]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available

implementations or their features. Readers are advised to note that other implementations may exist.

According to [RFC7942], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

Authors are requested to add a note to the RFC Editor at the top of this section, advising the Editor to remove the entire section before publication, as well as the reference to [RFC7942].

18.1. Linux XFRM

Linux

Organization: Linux kernel Project

Name: Linux Kernel https://www.kernel.org/

Description: Implements BEET mode in ESP. The initial support was added in 2006. It is widely used

Level of maturity: Stable used for over 15 years

Licensing: GPLv2

Implementation experience: There is no support for IPv4 fragments yet. IPv6 fragments appears to work. The BEE mode code is in production for over a decade

Contact: https://lore.kernel.org/netdev/

18.2. strongSwan

Organization: The strongSwan Project

Name: strongSwan https://docs.strongswan.org/docs/5.9/swanctl/swanctlConf.html

Description: Implement IKE negotiation and and ESP support for BEET mode Linux.

Level of maturity: Stable

Coverage: Implements negotiating BEET mode support in Child SA negotiations and using it in ESP. The initial support was added in 2006.

Licensing:
GPLv2

Implementation experience strongSwan use a private space notification value for IKE negotiation. USE_BEET_MODE (40961). As far we know BEET is widely used.

Contact Tobias Brunner tobias@strongswan.org

18.3. iproute2

Organization: The iproute2 Project

Name: iproute2 https://git.kernel.org/pub/scm/network/iproute2/iproute2.git

Description: Implements BEET mode support in ESP. e.g. command support "ip xfrm policy ... mode beet" . and "ip xfrm state .. mode beet". The initial support was added in 2006

Level of maturity: Stable

Licensing: GPLv2

Implementation experience: TBD

Contact: https://lore.kernel.org/netdev/ or Stephen Hemminger stephen@networkplumber.org

19. Acknowledgments

We sincerely thank the previous authors and contributors whose work laid the foundation for this document. Their insights and dedication have greatly influenced our work, also their contributions to the BEET mode implementation many years ago.

Special thanks to Peka Nikander for generously granting permission to use their previous work [I-D.nikander-esp-beet-mode].

We appreciate the guidance and support from Tero Kivinen in reaching out to previous authors during the document's development.

Our gratitude also goes to all those who provided feedback, guidance, and support throughout this document's development and operational experience. Your contributions were vital in making this work possible.

20. Normative References

[RFC2119]

Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <https://www.rfc-editor.org/info/rfc4301>.

[RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, DOI 10.17487/RFC4303, December 2005, <https://www.rfc-editor.org/info/rfc4303>.

[RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <https://www.rfc-editor.org/info/rfc7296>.

[RFC791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <https://www.rfc-editor.org/info/rfc791>.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

21. Informative References

[I-D.nikander-esp-beet-mode] Nikander, P. and J. Melen, "A Bound End-to-End Tunnel (BEET) mode for ESP", Work in Progress, Internet-Draft, draft-nikander-esp-beet-mode-09, 5 August 2008, <https://datatracker.ietf.org/doc/html/draft-nikander-esp-beet-mode-09>.

[RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, DOI 10.17487/RFC3552, July 2003, <https://www.rfc-editor.org/info/rfc3552>.

[RFC5201] Moskowitz, R., Nikander, P., Jokela, P., Ed., and T. Henderson, "Host Identity Protocol", RFC 5201, DOI 10.17487/RFC5201, April 2008, <https://www.rfc-editor.org/info/rfc5201>.

[RFC5202] Jokela, P., Moskowitz, R., and P. Nikander, "Using the Encapsulating Security Payload (ESP) Transport Format with the Host Identity Protocol (HIP)", RFC 5202, DOI 10.17487/RFC5202, April 2008, <https://www.rfc-editor.org/info/rfc5202>.

[RFC6982] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", RFC 6982, DOI 10.17487/RFC6982, July 2013, <https://www.rfc-editor.org/info/rfc6982>.

[RFC7401] Moskowitz, R., Ed., Heer, T., Jokela, P., and T. Henderson, "Host Identity Protocol Version 2 (HIPv2)", RFC 7401, DOI 10.17487/RFC7401, April 2015, <https://www.rfc-editor.org/info/rfc7401>.

[RFC7402] Jokela, P., Moskowitz, R., and J. Melen, "Using the Encapsulating Security Payload (ESP) Transport Format with the Host Identity Protocol (HIP)", RFC 7402, DOI 10.17487/RFC7402, April 2015, <https://www.rfc-editor.org/info/rfc7402>.

[RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <https://www.rfc-editor.org/info/rfc7942>.

[RFC9333] Migault, D. and T. Guggemos, "Minimal IP Encapsulating Security Payload (ESP)", RFC 9333, DOI 10.17487/RFC9333, January 2023, <https://www.rfc-editor.org/info/rfc9333>.

Appendix A. Additional Stuff

This becomes an Appendix.

Authors' Addresses

Antony Antony
secunet Security Networks AG

Email: antony.antony@secunet.com

Steffen Klassert
secunet Security Networks AG

Email: steffen.klassert@secunet.com