IPSECME Working Group
Internet-Draft
Updates: rfc7402 (if approved)
Intended status: Standards Track
Expires: 12 April 2026

R. Moskowitz, Ed.
HTT Consulting
P. Laari
Ericsson
J. Melen
Ericsson Software Technology
A. Antony
secunet
A. Gurtov
Link旦ping University

9 October 2025

A Bound End-to-End Tunnel (BEET) mode for ESP
draft-moskowitz-ipsecme-rfc7402-beet-update-02

Abstract

This document is an update to the Bound End-to-End Tunnel (BEET) mode for ESP in as described in [RFC7402]. It brings the description in alignment with the addition of BEET mode for IKEv2 without any processing changes as described in [RFC7402]

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 12 April 2026.

Copyright Notice

Table of Contents

1. Introduction

The Bound End-to-End Tunnel (BEET) mode for ESP [RFC4303], is an integral part of HIP [RFC7401]. BEET was initially defined in Appendix B of [RFC7402].

This document leaves the basic BEET processing unchanged to not impact any [RFC7402] implementations, but separately defines BEET for clearer referencing for use in IKEv2 [ikev2-beet-mode]. To assist with BEET in IPsec, this document includes Section 4.

2. Terms and Definitions

2.1. Requirements Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Protocol Definition

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

3.1. 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 Section 4.4.2 of [RFC4301].

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.

3.2. Packet Format

The wire packet format is identical to the ESP transport mode wire format as defined in Section 3.1.1 of [RFC4303]. 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 Section 5.1.2 of [RFC4301]. The following diagram illustrates ESP BEET mode positioning for typical IPv4 and IPv6 packets.

3.2.1. Inner IPv4 Datagram

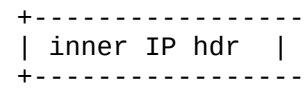

Figure 1: IPv4 INNER DATAGRAM BEFORE APPLYING ESP

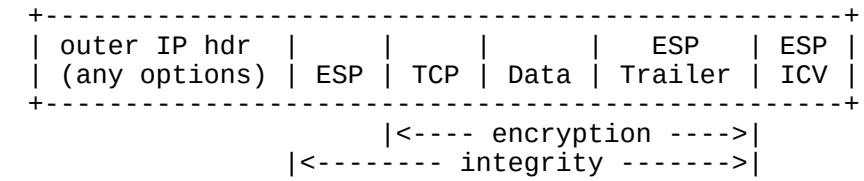

Figure 2: AFTER APPLYING ESP, OUTER v4 ADDRESSES

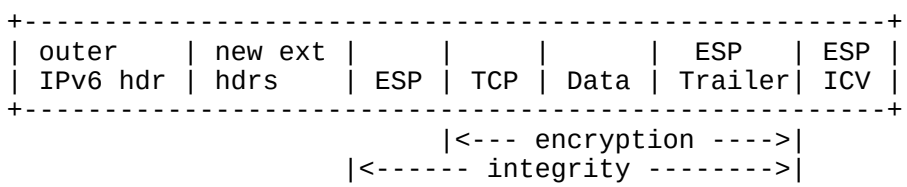

Figure 3: AFTER APPLYING ESP, OUTER v6 ADDRESSES

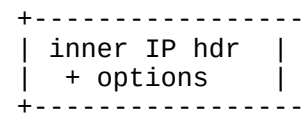

Figure 4: IPv4 INNER DATAGRAM with IP options BEFORE APPLYING ESP

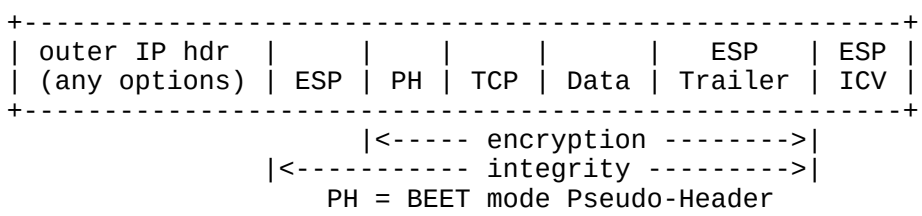

Figure 5: IPv4 AFTER APPLYING ESP, OUTER v4 ADDRESSES INNER IPv4 OPTIONS

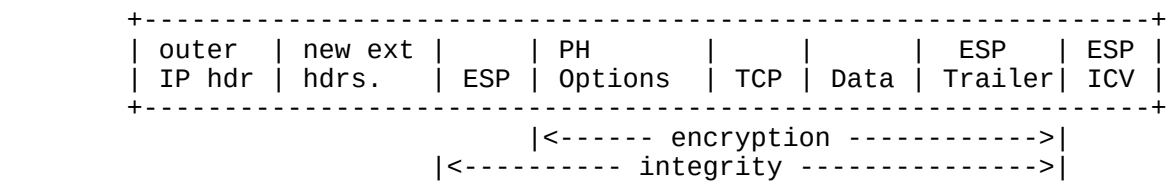

Figure 6: IPv4 + OPTIONS AFTER APPLYING ESP, OUTER IPv6 ADDRESSES

3.2.2. Inner IPv6 Datagram

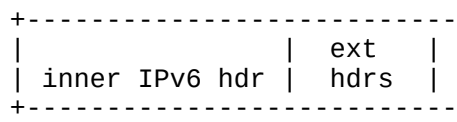

Figure 7: IPv6 DATAGRAM BEFORE APPLYING ESP

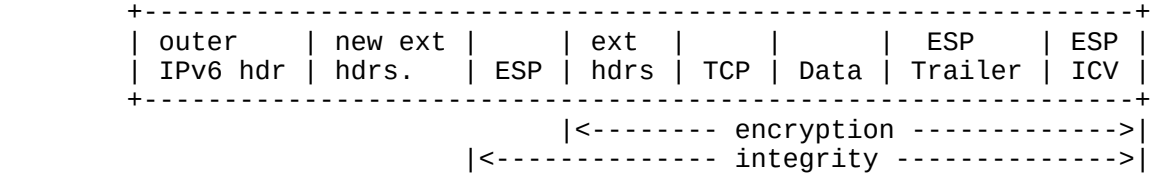

Figure 8: IPv6 DATAGRAM AFTER APPLYING ESP, OUTER IPv6 ADDRESSES

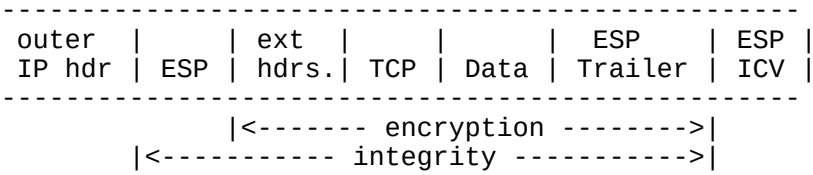

Figure 9: IPv6 DATAGRAM AFTER APPLYING ESP, OUTER IPv4 ADDRESSES

3.3. 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 3.4 and Section 3.5. 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 3.4 and Section 3.6

3.4. IP Header Processing

The biggest difference between 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 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 time of SA processing. 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 fields of the discarded inner header (except the IP addresses) to populate the fields of the new outer header. 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 BEET mode inner header is created, using the fields of the discarded outer header (except the IP addresses) to populate the fields of the new inner header. The IP addresses in the new header MUST be the inner addresses.

As the outer header fields are used as a hint for creating the inner header, it must be noted that the inner header differs as compared to a tunnel mode inner header. In BEET mode, the inner header will have the Time to Live (TTL), Don’t Fragment (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.

3.5. 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. If 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 Section 3.3 of [RFC4303].

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 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.

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 endpoints, the sender MUST encapsulate the options as defined Section 3.7.

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.

3.6. 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 Section 3.4 of [RFC4303]. 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 Header field to 94 and included the pseudo header as described in Section 3.7, 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 a 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?]

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.

3.7. Handling of IPv4 Options

In BEET mode, if IPv4 options are transported inside the tunnel, the sender MUST include a pseudo header after the ESP header. The pseudo header indicates that IPv4 options from the original packet are to be applied to the packet on the input side or the IPv4 fragmentation flags and fragmentation offset is set.

The sender MUST set the Next Header field in the ESP header to 94. The resulting pseudo header, including the IPv4 options, MUST be padded to an 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 a 4-octet boundary. The padding MUST be filled with No Operation (NOP) options as defined in (Internet Header Format) Section 3.1 of [RFC791] (Internet Protocol). 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-octet units.

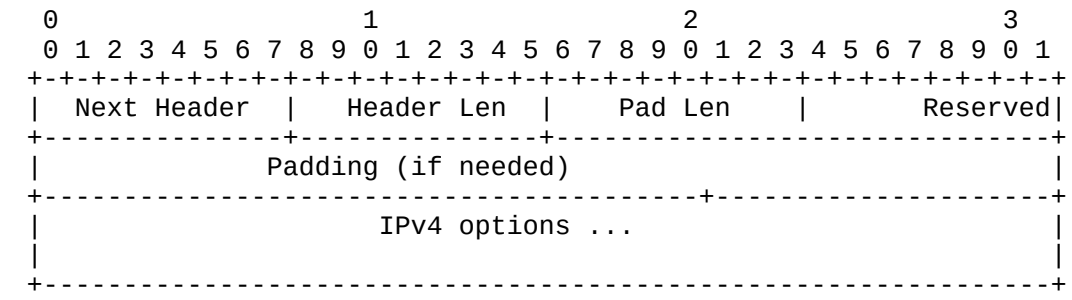

Figure 10: BEET mode pseudo header format

Next Header identifies the data following this header.

Length in octets 8-bit unsigned integer. Length of the pseudo header in 8-octet units, not including the first 8 octets.

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.

[AA NOTE: Note: when the IPv4 options are present, the outer header's IHL would be different from the inner header IHL NEXT paragraph is extra???]

3.8. Handling of Inner IP Fragment

3.8.1. Inner IPv4 Fragment

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, Figure 11.

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 Section 3.1 of [RFC791] 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.

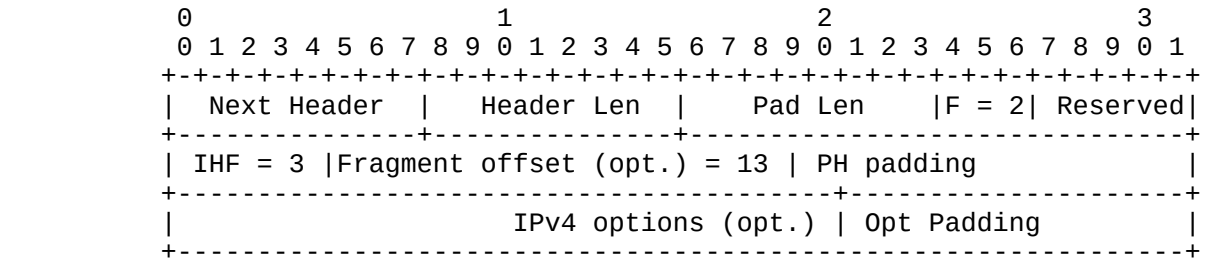

Figure 11: BEET mode pseudo header format

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.

3.8.2. Inner IPv6 Fragment

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

3.8.3. IPv4-in-IPv6

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 3.8.1.

3.8.4. IPv6-in-IPv4

Mixed family IPv6 inside and IPv4 outside. When the inner datagram is an IPv6 datagram with IPv6 optional extension headers, copy it into ESP payload Section 3.8.2

4. IPsec Specific Policy Considerations

In this section we describe how BEET mode affects 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.

5. Security Considerations

In this document, the usage of ESP [RFC4303] between hosts to protect data traffic is introduced. The security considerations for ESP are discussed in the ESP specification.

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.

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.

6. IANA Considerations

NONE

7. Implementation Status

BEET mode was first implemented as a patch to Linux kernel by Helsinki Institute for Information Technology, Finland around 2007. It was later submitted for kernel integration and included into mainstream release around 2010. See BEET patch for 2.6.20.21 linux kernel (Joakim Koskela). [HIPL] (HIP for Linux) is a software project that implements the Host Identity Protocol (HIP) on Linux systems. It was funded under InfraHIP label by Finnish innovation agency Tekes.

[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].

8. Acknowledgments

Antony Antony performed the original extraction of Appendix B of [RFC7402] to form this draft. Most of the work after that has been cleaning up the formatting and adding more IPsec specific content.

9. 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>.

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

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

10. Informative References

[HIPL] Helsinki Institute for Information Technology, "HIP for Linux", <https://mkomu.kapsi.fi/hipl/>.

[ikev2-beet-mode] Antony, A. and S. Klassert, "IKEv2 negotiation for Bound End-to-End Tunnel (BEET) mode ESP", Work in Progress,

Internet-Draft, draft-ietf-ipsecme-ikev2-beet-mode-01, 16 September 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-ipsecme-ikev2-beet-mode-01>.

[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>.

[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>.

[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>.

Authors' Addresses

Robert Moskowitz (editor)
HTT Consulting
Oak Park, MI 48237
United States of America
Email: rgm@labs.htt-consult.com

Petri Laari
Ericsson
Hirsalantie 11
FI-02420 JORVAS
Finland
Email: petri.laari@ericsson.com

Jan Melen
Ericsson Software Technology
Hirsalantie 11
FI-02420 JORVAS
Finland
Email: Jan.Melen@ericsson.com

Antony Antony
secunet Security Networks AG
Email: antony.antony@secunet.com

Andrei Gurtov
Link旦ping University
IDA
SE-58183 Link旦ping
Sweden
Email: gurtov@acm.org