Network Working Group
Internet-Draft
Obsoletes: rfc5320 (if approved)
Updates: rfc2460 (if approved)
Intended status: Standards Track
Expires: April 24, 2014

F. Templin, Ed.
Boeing Research & Technology
October 21, 2013

The Subnetwork Encapsulation and Adaptation Layer (SEAL) draft-templin-intarea-seal-65.txt

Abstract

This document specifies a Subnetwork Encapsulation and Adaptation Layer (SEAL). SEAL operates over virtual topologies configured over connected IP network routing regions bounded by encapsulating border nodes. These virtual topologies are manifested by tunnels that may span multiple IP and/or sub-IP layer forwarding hops, where they may incur packet duplication, packet reordering, source address spoofing and traversal of links with diverse Maximum Transmission Units (MTUs). SEAL addresses these issues through the encapsulation and messaging mechanisms specified in this document.

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 http://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 April 24, 2014.

Copyright Notice

Table of Contents

1. Introduction

As Internet technology and communication has grown and matured, many techniques have developed that use virtual topologies (manifested by tunnels of one form or another) over an actual network that supports the Internet Protocol (IP) [RFC0791][RFC2460]. Those virtual topologies have elements that appear as one network layer hop, but are actually multiple IP or sub-IP layer hops. These multiple hops often have quite diverse properties that are often not even visible to the endpoints of the virtual hop. This introduces failure modes that are not dealt with well in current approaches.

The use of IP encapsulation (also known as "tunneling") has long been considered as the means for creating such virtual topologies (e.g., see [RFC2003][RFC2473]). Tunnels serve a wide variety of purposes, including mobility, security, routing control, traffic engineering, multihoming, etc., and will remain an integral part of the architecture moving forward. However, the encapsulation headers often include insufficiently provisioned per-packet identification values. IP encapsulation also allows an attacker to produce encapsulated packets with spoofed source addresses even if the source address in the encapsulating header cannot be spoofed. A denial-of-service vector that is not possible in non-tunneled subnetworks is therefore presented.

Additionally, the insertion of an outer IP header reduces the effective path MTU visible to the inner network layer. When IPv6 is used as the encapsulation protocol, original sources expect to be informed of the MTU limitation through IPv6 Path MTU discovery (PMTUD) [RFC1981]. When IPv4 is used, this reduced MTU can be accommodated through the use of IPv4 fragmentation, but unmitigated in-the-network fragmentation has been found to be harmful through operational experience and studies conducted over the course of many years [FRAG][FOLK][RFC4963]. Additionally, classical IPv4 PMTUD [RFC1191] has known operational issues that are exacerbated by in-the-network tunnels [RFC2923][RFC4459].

The following subsections present further details on the motivation and approach for addressing these issues.

1.1. Motivation

Before discussing the approach, it is necessary to first understand the problems. In both the Internet and private-use networks today, IP is ubiquitously deployed as the Layer 3 protocol. The primary functions of IP are to provide for routing, addressing, and a fragmentation and reassembly capability used to accommodate links with diverse MTUs. While it is well known that the IP address space

is rapidly becoming depleted, there is also a growing awareness that other IP protocol limitations have already or may soon become problematic.

First, the Internet historically provided no means for discerning whether the source addresses of IP packets are authentic. This shortcoming is being addressed more and more through the deployment of site border router ingress filters [RFC2827], however the use of encapsulation provides a vector for an attacker to circumvent filtering for the encapsulated packet even if filtering is correctly applied to the encapsulation header. Secondly, the IP header does not include a well-behaved identification value unless the source has included a fragment header for IPv6 or unless the source permits fragmentation for IPv4. These limitations preclude an efficient means for routers to detect duplicate packets and packets that have been re-ordered within the subnetwork. Additionally, recent studies have shown that the arrival of fragments at high data rates can cause denial-of-service (DoS) attacks on performance-sensitive networking gear, prompting some administrators to configure their equipment to drop fragments unconditionally [I-D.taylor-v6ops-fragdrop].

For IPv4 encapsulation, when fragmentation is permitted the header includes a 16-bit Identification field, meaning that at most 2^16 unique packets with the same (source, destination, protocol)-tuple can be active in the network at the same time [RFC6864]. (When middleboxes such as Network Address Translators (NATs) re-write the Identification field to random values, the number of unique packets is even further reduced.) Due to the escalating deployment of high-speed links, however, these numbers have become too small by several orders of magnitude for high data rate packet sources such as tunnel endpoints [RFC4963].

Furthermore, there are many well-known limitations pertaining to IPv4 fragmentation and reassembly - even to the point that it has been deemed "harmful" in both classic and modern-day studies (see above). In particular, IPv4 fragmentation raises issues ranging from minor annoyances (e.g., in-the-network router fragmentation [RFC1981]) to the potential for major integrity issues (e.g., mis-association of the fragments of multiple IP packets during reassembly [RFC4963]).

As a result of these perceived limitations, a fragmentation-avoiding technique for discovering the MTU of the forward path from a source to a destination node was devised through the deliberations of the Path MTU Discovery Working Group (MTUDWG) during the late 1980's through early 1990's which resulted in the publication of [RFC1191]. In this negative feedback-based method, the source node provides explicit instructions to routers in the path to discard the packet and return an ICMP error message if an MTU restriction is

encountered. However, this approach has several serious shortcomings that lead to an overall "brittleness" [RFC2923].

In particular, site border routers in the Internet have been known to discard ICMP error messages coming from the outside world. This is due in large part to the fact that malicious spoofing of error messages in the Internet is trivial since there is no way to authenticate the source of the messages [RFC5927]. Furthermore, when a source node that requires ICMP error message feedback when a packet is dropped due to an MTU restriction does not receive the messages, a path MTU-related black hole occurs. This means that the source will continue to send packets that are too large and never receive an indication from the network that they are being discarded. This behavior has been confirmed through documented studies showing clear evidence of PMTUD failures for both IPv4 and IPv6 in the Internet today [TBIT][WAND][SIGCOMM][RIPE].

The issues with both IP fragmentation and this "classical" PMTUD method are exacerbated further when IP tunneling is used [RFC4459]. For example, an ingress tunnel endpoint (ITE) may be required to forward encapsulated packets into the subnetwork on behalf of hundreds, thousands, or even more original sources. If the ITE allows IP fragmentation on the encapsulated packets, persistent fragmentation could lead to undetected data corruption due to Identification field wrapping and/or reassembly congestion at the ETE. If the ITE instead uses classical IP PMTUD it must rely on ICMP error messages coming from the subnetwork that may be suspect, subject to loss due to filtering middleboxes, or insufficiently provisioned for translation into error messages to be returned to the original sources.

Although recent works have led to the development of a positive feedback-based end-to-end MTU determination scheme [RFC4821], they do not excuse tunnels from accounting for the encapsulation overhead they add to packets. Moreover, in current practice existing tunneling protocols mask the MTU issues by selecting a "lowest common denominator" MTU that may be much smaller than necessary for most paths and difficult to change at a later date. Therefore, a new approach to accommodate tunnels over links with diverse MTUs is necessary.

1.2. Approach

This document concerns subnetworks manifested through a virtual topology configured over a connected network routing region and bounded by encapsulating border nodes. Example connected network routing regions include Mobile Ad hoc Networks (MANETs), enterprise networks and the global public Internet itself. Subnetwork border

nodes forward unicast and multicast packets over the virtual topology across multiple IP and/or sub-IP layer forwarding hops that may introduce packet duplication and/or traverse links with diverse Maximum Transmission Units (MTUs).

This document introduces a Subnetwork Encapsulation and Adaptation Layer (SEAL) for tunneling inner network layer protocol packets over IP subnetworks that connect Ingress and Egress Tunnel Endpoints (ITEs/ETEs) of border nodes. It provides a modular specification designed to be tailored to specific associated tunneling protocols. (A transport-mode of operation is also possible but out of scope for this document.)

SEAL provides a mid-layer encapsulation that accommodates links with diverse MTUs, and allows routers in the subnetwork to perform efficient duplicate packet and packet reordering detection. The encapsulation further ensures message origin authentication, packet header integrity and anti-replay in environments in which these functions are necessary.

SEAL treats tunnels that traverse the subnetwork as ordinary links that must support network layer services. Moreover, SEAL provides dynamic mechanisms (including limited segmentation and reassembly) to ensure a maximal path MTU over the tunnel. This is in contrast to static approaches which avoid MTU issues by selecting a lowest common denominator MTU value that may be overly conservative for the vast majority of tunnel paths and difficult to change even when larger MTUs become available.

1.3. Differences with RFC5320

This specification of SEAL is descended from an experimental independent RFC publication of the same name [RFC5320]. However, this specification introduces a number of fundamental differences from the earlier publication. This specification therefore obsoletes (i.e., and does not update) [RFC5320].

First, this specification includes a protocol version field in the SEAL header whereas [RFC5320] does not, and therefore cannot be updated by future revisions. Secondly, [RFC5320] forms a 32-bit Identification value by concatenating the 16-bit IPv4 Identification field with a 16-bit Identification "extension" field in the SEAL header. This means that [RFC5320] can only operate over IPv4 networks (since IPv6 headers do not include a 16-bit version number) and that the SEAL Identification value can be corrupted if the Identification in the outer IPv4 header is rewritten. In contrast, this specification includes a 32-bit Identification value that is independent of any identification fields found in the inner or outer

IP headers, and is therefore compatible with any inner and outer IP protocol version combinations.

Additionally, the SEAL segmentation and reassembly procedures defined in [RFC5320] differ significantly from those found in this specification. In particular, this specification defines an 13-bit Offset field that allows for finer-grained segment sizes when SEAL segmentation is necessary. In contrast, [RFC5320] includes only a 3-bit Segment field and performs reassembly through concatenation of consecutive segments.

This version of SEAL also includes an optional Integrity Check Vector (ICV) that can be used to digitally sign the SEAL header and the leading portion of the encapsulated inner packet. This allows for a lightweight integrity check and a loose message origin authentication capability. The header further includes new control bits as well as a link identification field for additional control capabilities.

Finally, this version of SEAL includes a new messaging protocol known as the SEAL Control Message Protocol (SCMP), whereas [RFC5320] performs signalling through the use of SEAL-encapsulated ICMP messages. The use of SCMP allows SEAL-specific departures from ICMP, as well as a control messaging capability that extends to other specifications, including Virtual Enterprise Traversal (VET) [I-D.templin-intarea-vet].

2. Terminology

The following terms are defined within the scope of this document:

subnetwork
: a virtual topology configured over a connected network routing region and bounded by encapsulating border nodes.

IP
: used to generically refer to either Internet Protocol (IP) version, i.e., IPv4 or IPv6.

Ingress Tunnel Endpoint (ITE)
: a portal over which an encapsulating border node (host or router) sends encapsulated packets into the subnetwork.

Egress Tunnel Endpoint (ETE)
: a portal over which an encapsulating border node (host or router) receives encapsulated packets from the subnetwork.

SEAL Path
a subnetwork path from an ITE to an ETE beginning with an underlying link of the ITE as the first hop. Note that, if the ITE's interface connection to the underlying link assigns multiple IP addresses, each address represents a separate SEAL path.

inner packet
an unencapsulated network layer protocol packet (e.g., IPv4 [RFC0791], OSI/CLNP [RFC0994], IPv6 [RFC2460], etc.) before any outer encapsulations are added. Internet protocol numbers that identify inner packets are found in the IANA Internet Protocol registry [RFC3232]. SEAL protocol packets that incur an additional layer of SEAL encapsulation are also considered inner packets.

outer IP packet
a packet resulting from adding an outer IP header (and possibly other outer headers) to a SEAL-encapsulated inner packet.

packet-in-error
the leading portion of an invoking data packet encapsulated in the body of an error control message (e.g., an ICMPv4 [RFC0792] error message, an ICMPv6 [RFC4443] error message, etc.).

Packet Too Big (PTB) message
a control plane message indicating an MTU restriction (e.g., an ICMPv6 "Packet Too Big" message [RFC4443], an ICMPv4 "Fragmentation Needed" message [RFC0792], etc.).

Don't Fragment (DF) bit
a bit that indicates whether the packet may be fragmented by the network. The DF bit is explicitly included in the IPv4 header [RFC0791] and may be set to '0' to allow fragmentation or '1' to disallow further in-network fragmentation. The bit is absent from the IPv6 header [RFC2460], but implicitly set to '1' because fragmentation can occur only at IPv6 sources.

The following abbreviations correspond to terms used within this document and/or elsewhere in common Internetworking nomenclature:

HLEN - the length of the SEAL header plus outer headers

ICV - Integrity Check Vector

MAC - Message Authentication Code

MTU - Maximum Transmission Unit

SCMP - the SEAL Control Message Protocol

SDU - SCMP Destination Unreachable message

SPP - SCMP Parameter Problem message

SPTB - SCMP Packet Too Big message

SEAL - Subnetwork Encapsulation and Adaptation Layer

TE - Tunnel Endpoint (i.e., either ingress or egress)

VET - Virtual Enterprise Traversal

3. Requirements

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]. When used in lower case (e.g., must, must not, etc.), these words MUST NOT be interpreted as described in [RFC2119], but are rather interpreted as they would be in common English.

4. Applicability Statement

SEAL was originally motivated by the specific case of subnetwork abstraction for Mobile Ad hoc Networks (MANETs), however the domain of applicability also extends to subnetwork abstractions over enterprise networks, mobile networks, ISP networks, SO/HO networks, the global public Internet itself, and any other connected network routing region.

SEAL provides a network sublayer for encapsulation of an inner network layer packet within outer encapsulating headers. SEAL can also be used as a sublayer within a transport layer protocol data payload, where transport layer encapsulation is typically used for Network Address Translator (NAT) traversal as well as operation over subnetworks that give preferential treatment to certain "core" Internet protocols, e.g., TCP, UDP, etc. (However, note that TCP encapsulation may not be appropriate for all use cases; particularly those that require low delay and/or delay variance.) The SEAL header is processed in the same manner as for IPv6 extension headers, i.e., it is not part of the outer IP header but rather allows for the creation of an arbitrarily extensible chain of headers in the same way that IPv6 does.

To accommodate MTU diversity, the Ingress Tunnel Endpoint (ITE) may need to perform limited segmentation which the Egress Tunnel Endpoint (ETE) reassembles. The ETE further acts as a passive observer that informs the ITE of any packet size limitations. This allows the ITE to return appropriate PMTUD feedback even if the network path between the ITE and ETE filters ICMP messages.

SEAL further provides mechanisms to ensure message origin authentication, packet header integrity, and anti-replay. The SEAL framework is therefore similar to the IP Security (IPsec) Authentication Header (AH) [RFC4301][RFC4302], however it provides only minimal hop-by-hop authenticating services while leaving full data integrity, authentication and confidentiality services as an end-to-end consideration.

In many aspects, SEAL also very closely resembles the Generic Routing Encapsulation (GRE) framework [RFC1701]. SEAL can therefore be applied in the same use cases that are traditionally addressed by GRE, but goes beyond GRE to also provide additional capabilities (e.,g., path MTU accommodation, message origin authentication, etc.) as described in this document. The SEAL header is also exactly analogous to the IPv6 Fragment Header, and in fact shares the same format. SEAL can therefore re-use most existing code that implements IPv6 fragmentation and reassembly.

In practice, SEAL is typically used as an encapsulation sublayer in conjunction with existing tunnel types such as IPsec, GRE, IP-in-IPv6 [RFC2473], IP-in-IPv4 [RFC4213][RFC2003], etc. When used with existing tunnel types that insert mid-layer headers between the inner and outer IP headers (e.g., IPsec, GRE, etc.), the SEAL header is inserted between the mid-layer headers and outer IP header.

5. SEAL Specification

The following sections specify the operation of SEAL:

5.1. SEAL Tunnel Model

SEAL is an encapsulation sublayer used within point-to-point, point-to-multipoint, and non-broadcast, multiple access (NBMA) tunnels. Each SEAL path is configured over one or more underlying interfaces attached to subnetwork links. The SEAL tunnel connects an ITE to one or more ETE "neighbors" via encapsulation across an underlying subnetwork, where the tunnel neighbor relationship may be bidirectional, partially unidirectional or fully unidirectional.

A bidirectional tunnel neighbor relationship is one over which both

TEs can exchange both data and control messages. A partially unidirectional tunnel neighbor relationship allows the near end ITE to send data packets forward to the far end ETE, while the far end only returns control messages when necessary. Finally, a fully unidirectional mode of operation is one in which the near end ITE can receive neither data nor control messages from the far end ETE.

Implications of the SEAL bidirectional and unidirectional models are the same as discussed in [I-D.templin-intarea-vet].

5.2. SEAL Model of Operation

SEAL-enabled ITEs encapsulate each inner packet in any ancillary tunnel protocol headers, a SEAL header, any outer header encapsulations and in some instances a SEAL trailer as shown in Figure 1:

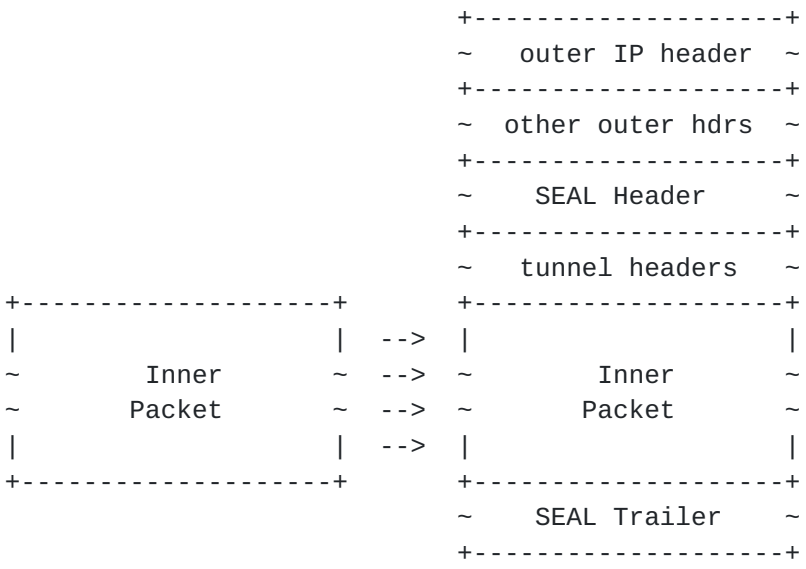


Figure 1: SEAL Encapsulation

The ITE inserts the SEAL header according to the specific tunneling protocol. For simple encapsulation of an inner network layer packet within an outer IP header, the ITE inserts the SEAL header following the outer IP header and before the inner packet as: IP/SEAL/{inner packet}.

For encapsulations over transports such as UDP, the ITE inserts the SEAL header following the outer transport layer header and before the inner packet, e.g., as IP/UDP/SEAL/{inner packet}. In that case, the UDP header is seen as an "other outer header" as depicted in Figure 1 and the outer IP and transport layer headers are together seen as the outer encapsulation headers. (Note that outer transport layer

headers such as UDP must sometimes be included to ensure that SEAL packets will traverse the path to the ETE without loss due filtering middleboxes. The ETE MUST accept both IP/SEAL and IP/UDP/SEAL as equivalent packets so that the ITE can discontinue outer transport layer encapsulation if the path supports raw IP/SEAL encapsulation.)

For SEAL encapsulations that involve tunnel types that include ancillary tunnel headers (e.g., GRE, IPsec, etc.) the ITE inserts the SEAL header as a leading extension to the tunnel headers, i.e., the SEAL encapsulation appears as part of the same tunnel and not a separate tunnel. For example, for GRE the ITE iserts the SEAL header as IP/SEAL/GRE/{inner packet}, and for IPsec the ITE inserts the SEAL header as IP/SEAL/IPsec-header/{inner packet}/IPsec-trailer. In such cases, SEAL considers the length of the inner packet only (i.e., and not the other tunnel headers and trailers) when performing its packet size calculations.

SEAL supports both "nested" tunneling and "re-encapsulating" tunneling. Nested tunneling occurs when a first tunnel is encapsulated within a second tunnel, which may then further be encapsulated within additional tunnels. Nested tunneling can be useful, and stands in contrast to "recursive" tunneling which is an anomalous condition incurred due to misconfiguration or a routing loop. Considerations for nested tunneling and avoiding recursive tunneling are discussed in Section 4 of [RFC2473] as well as in Section 9 of this document.

Re-encapsulating tunneling occurs when a packet arrives at a first ETE, which then acts as an ITE to re-encapsulate and forward the packet to a second ETE connected to the same subnetwork. In that case each ITE/ETE transition represents a segment of a bridged path between the ITE nearest the source and the ETE nearest the destination. Considerations for re-encapsulating tunneling are discussed in[I-D.templin-ironbis]. Combinations of nested and re-encapsulating tunneling are also naturally supported by SEAL.

The SEAL ITE considers each underlying interface as the ingress attachment point to a separate SEAL path to the ETE. The ITE therefore may experience different path MTUs on different SEAL paths.

Finally, the SEAL ITE ensures that the inner network layer protocol will see a minimum MTU of 1500 bytes over each SEAL path regardless of the outer network layer protocol version, i.e., even if a small amount of segmentation and reassembly are necessary. This is to avoid path MTU "black holes" for the minimum MTU configured by the vast majority of links in the Internet. Note that in some scenarios, however, reassembly may place a heavy burden on the ETE. In that case, the ITE can avoid invoking segmentation and instead report an

MTU smaller than 1500 bytes to the original source.

5.3. SEAL Encapsulation Format

SEAL encapsulates each inner packet within any ancillary tunneling protocol headers and a SEAL header. The SEAL header shares the same format as the IPv6 Fragment Header [RFC2460] and is identified by the same IP protocol number assigned for the IPv6 Fragment Header (type '44') [I-D.ietf-6man-ext-transmit]. The SEAL header is differentiated from the IPv6 Fragment Header by including a non-zero value in the most significant two bits of the IPv6 Fragment Header "Reserved" field; these two bits will heretofore serve as a SEAL protocol version number. SEAL therefore updates the IPv6 Fragment Header specification found in [RFC2460].

The SEAL header is formatted as shown in Figure 2:

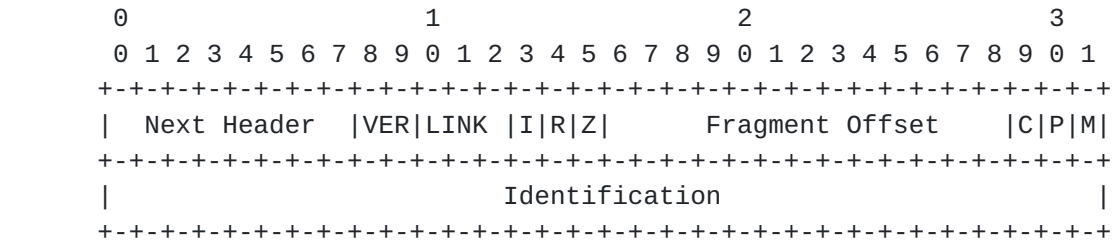


Figure 2: SEAL Encapsulation Format

The fields of the SEAL header are formatted as follows:

Next Header (8) an 8-bit field that encodes the next header Internet Protocol number the same as for the IPv4 protocol and IPv6 next header fields.

VER (2)
a 2-bit version field. This document specifies Version 1 of the SEAL protocol, i.e., the VER field encodes the value '01'.

LINK (3)
a 3-bit link identification value, set to a unique value by the ITE for each SEAL path over which it will send encapsulated packets to the ETE (up to 8 SEAL paths per ETE are therefore supported). Note that, if the ITE's interface connection to the underlying link assigns multiple IP addresses, each address represents a separate SEAL path that must be assigned a separate link ID.

I (1)
the "Integrity Check Vector (ICV) included" bit.

R (1)
the "Redirects Permitted" bit when used by VET (see: [I-D.templin-intarea-vet]); reserved for future use in other contexts.

Z (1)
a 1-bit Reserved field. Initialized to zero for transmission; ignored on reception.

Fragment Offset (13) a 13-bit Offset field. The offset, in 8-octet units, of the data following this header.

C (1)
the "Control/Data" bit. Set to 1 by the ITE in SEAL Control Message Protocol (SCMP) control messages, and set to 0 in ordinary data packets.

P (1)
The "Probe" bit when C=0; set to 1 by the ITE in SEAL probe data packets for which it wishes to receive an explicit acknowledgement from the ETE. The "Pass" bit when C=1; set to 1 by the ETE in SCMP messages it relays to the ITE on behalf of another SEAL path.

M (1) the "More Segments" bit. Set to 1 in a non-final segment and set to 0 in the final segment of the SEAL packet.

Identification (32)
a 32-bit per-packet identification field. Set to a randomly-initialized 32-bit value that is monotonically-incremented for each SEAL packet transmitted to this ETE.

When an IIntegrity Check Vector (ICV) is included, it is added as a trailing field at the end of the SEAL packet. The ICV is formatted as shown in Figure 3:

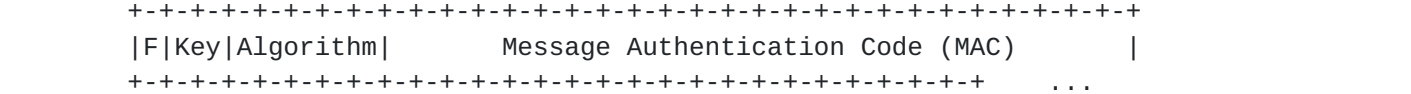


Figure 3: Integrity Check Vector (ICV) Format

As shown in the figure, the ICV begins with a 1-octet control field with a 1-bit (F)lag, a 2-bit Key identifier and a 5-bit Algorithm identifier. The control octet is followed by a variable-length Message Authentication Code (MAC). The ITE maintains a per ETE

algorithm and secret key to calculate the MAC in each packet it will send to this ETE. (By default, the ITE sets the F bit and Algorithm fields to 0 to indicate use of the HMAC-SHA-1 algorithm with a 160 bit shared secret key to calculate an 80 bit MAC per [RFC2104] over the leading 128 bytes of the packet. Other values for F and Algorithm are out of scope.)

5.4. ITE Specification

5.4.1. Tunnel MTU

The tunnel must present a stable MTU value to the inner network layer as the size for admission of inner packets into the tunnel. Since tunnels may support a large set of SEAL paths that accept widely varying maximum packet sizes, however, a number of factors should be taken into consideration when selecting a tunnel MTU.

Due to the ubiquitous deployment of standard Ethernet and similar networking gear, the nominal Internet cell size has become 1500 bytes; this is the de facto size that end systems have come to expect will either be delivered by the network without loss due to an MTU restriction on the path or a suitable ICMP Packet Too Big (PTB) message returned. When large packets sent by end systems incur additional encapsulation at an ITE, however, they may be dropped silently within the tunnel since the network may not always deliver the necessary PTBs [RFC2923]. The ITE SHOULD therefore set a tunnel MTU of at least 1500 bytes and provide accommodations to ensure that packets up to that size are successfully conveyed to the ETE.

The inner network layer protocol consults the tunnel MTU when admitting a packet into the tunnel. For non-SEAL inner IPv4 packets with the IPv4 Don't Fragment (DF) bit cleared (i.e, DF==0), if the packet is larger than the tunnel MTU the inner IPv4 layer uses IPv4 fragmentation to break the packet into fragments no larger than the MTU. The ITE then admits each fragment into the tunel as an independent packet.

For all other inner packets, the inner network layer admits the packet if it is no larger than the tunnel MTU; otherwise, it drops the packet and sends a PTB error message to the source with the MTU value set to the MTU. The message contains as much of the invoking packet as possible without the entire message exceeding the network layer minimum MTU size.

The ITE can alternatively set an indefinite tunnel MTU such that all inner packets are admitted into the tunnel regardless of their size (theoretical maximums are 64KB for IPv4 and 4GB for IPv6 [RFC2675]). For ITEs that host applications that use the tunnel directly, this

option must be carefully coordinated with protocol stack upper layers since some upper layer protocols (e.g., TCP) derive their packet sizing parameters from the MTU of the outgoing interface and as such may select too large an initial size. This is not a problem for upper layers that use conservative initial maximum segment size estimates and/or when the tunnel can reduce the upper layer's maximum segment size, e.g., by reducing the size advertised in the MSS option of outgoing TCP messages (sometimes known as "MSS clamping").

In light of the above considerations, the ITE SHOULD configure an indefinite MTU on *router* tunnels so that SEAL performs all subnetwork adaptation from within the tunnel as specified in the following sections. The ITE MAY instead set a smaller MTU on *host* tunnels; in that case, the RECOMMENDED MTU is the maximum of 1500 bytes and the smallest MTU among all of the underlying links minus the size of the encapsulation headers.

5.4.2. Tunnel Neighbor Soft State

The ITE maintains a number of soft state variables for each ETE and for each SEAL path.

The ITE maintains a per ETE window of Identification values for the packets it has recently sent to this ETE as welll as a per ETE window of Identification values for the packets it has recently received from this ETE. The ITE then includes an Identification in each packet it sends to this ETE.

When message origin authentication and integrity checking is required, the ITE sets a variable "USE_ICV" to TRUE, and includes a trailing ICV in each packet it sends to this ETE; otherwise, it sets USE_ICV to FALSE.

For each SEAL path, the ITE must also account for encapsulation header lengths. The ITE therefore maintains the per SEAL path constant values "SHLEN" set to the length of the SEAL header and trailer, "THLEN" set to the length of the outer encapsulating transport layer headers (or 0 if outer transport layer encapsulation is not used), "IHLEN" set to the length of the outer IP layer header, and "HLEN" set to (SHLEN+THLEN+IHLEN). (The ITE must include the length of the uncompressed headers even if header compression is enabled when calculating these lengths.) When SEAL is used in conjunction with another tunnel type such as GRE or IPsec, the length of the headers associated with those tunnels is also included in the HLEN calculation for the first segment only and the length of the associated trailers is included in the HLEN calculation for the final segment only.

The ITE maintains a per SEAL path variable "MAXMTU" initialized to the maximum of (1500+HLEN) bytes and the MTU of the underlying link. The ITE further sets a variable 'MINMTU' to the minimum MTU for the SEAL path over which encapsulated packets will travel. For IPv6 paths, the ITE sets MINMTU=1280 per [RFC2460]. For IPv4 paths, the ITE sets MINMTU=576 based on practical interpretation of [RFC1122] even though the theoretical MINMTU for IPv4 is only 68 bytes [RFC0791].

The ITE can also set MINMTU to a larger value if there is reason to believe that the minimum path MTU is larger, or to a smaller value if there is reason to believe the MTU is smaller, e.g., if there may be additional encapsulations on the path. If this value proves too large, the ITE will receive PTB message feedback either from the ETE or from a router on the path and will be able to reduce its MINMTU to a smaller value. (Note that since IPv4 links with MTUs smaller than 1280 are presumably peformance-constrained, the ITE can instead initialize MINMTU to 1280 the same as for IPv6. If this value proves too large, standard IPv4 fragmentation and reassembly will provide short term accommodation for the sizing constraints while the ITE readjusts its MINMTU estimate.)

The ITE may instead maintain the packet sizing variables and constants as per ETE (rather than per SEAL path) values. In that case, the values reflect the smallest MTU size across all of the SEAL paths associated with this ETE.

5.4.3. SEAL Layer Pre-Processing

The SEAL layer is logically positioned between the inner and outer network protocol layers, where the inner layer is seen as the (true) network layer and the outer layer is seen as the (virtual) data link layer. Each packet to be processed by the SEAL layer is either admitted into the tunnel by the inner network layer protocol as described in Section 5.4.1 or is undergoing re-encapsulation from within the tunnel. The SEAL layer sees the former class of packets as inner packets that include inner network and transport layer headers, and sees the latter class of packets as transitional SEAL packets that include the outer and SEAL layer headers that were inserted by the previous hop SEAL ITE. For these transitional packets, the SEAL layer re-encapsulates the packet with new outer and SEAL layer headers when it forwards the packet to the next hop SEAL ITE.

We now discuss the SEAL layer pre-processing actions for these two classes of packets.

5.4.3.1. Inner Packet Pre-Processing

For each for non-SEAL IPv4 inner packet with DF==0 in the IP header and IPv6 inner packet with a fragment header and with (MF=0; Offset=0), if the packet is larger than (MINMTU-HLEN) the ITE uses IP fragmentation to fragment the packet into N pieces, where N is minimized. (For IPv6 as the inner protocol, the first fragment MUST be at least as large as the IPv6 minimum of 1280 bytes so that the entire IPv6 header chain is likely to fit within the first segment.) The ITE then submits each fragment for SEAL encapsulation as specified in Section 5.4.4.

For all other inner packets, if the packet is no larger than (MAXMTU-HLEN) for the corresponding SEAL path the ITE submits it for SEAL encapsulation as specified in Section 5.4.4. Otherwise, the ITE drops the packet and sends an ordinary PTB message appropriate to the inner protocol version (subject to rate limiting) with the MTU field set to (MAXMTU-HLEN). (For IPv4 SEAL packets with DF==0, the ITE SHOULD set DF=1 and re-calculate the IPv4 header checksum before generating the PTB message in order to avoid bogon filters.) After sending the PTB message, the ITE discards the inner packet.

5.4.3.2. Transitional SEAL Packet Pre-Processing

For each transitional packet that is to be processed by the SEAL layer from within the tunnel, if the packet is larger than MAXMTU bytes for the next hop SEAL path the ITE sends an SCMP Packet Too Big (SPTB) message to the previous hop subject to rate limiting with the MTU field set to MAXMTU and with (C=1; P=1) in the SEAL header (see: Section 5.6.1.1). After sending the SPTB message, the ITE discards the packet. Otherwise, the ITE sets aside the encapsulating SEAL and outer headers and submits the inner packet for SEAL re-encapsulation as specified in Section 5.4.4. (Note that in the calculation for MAXMTU, HLEN for the next hop SEAL path may be different than HLEN for the previous hop. In that case, MAXMTU must reflect the smaller of the two HLEN values.)

5.4.4. SEAL Encapsulation and Segmentation

For each inner packet/fragment submitted for SEAL encapsulation, the ITE next encapsulates the packet in a SEAL header formatted as specified in Section 5.3. The ITE next sets (C=0; P=0), sets LINK to the value assigned to the underlying SEAL path, and sets the Next Header field to the protocol number corresponding to the address family of the encapsulated inner packet. For example, the ITE sets the Next Header field to the value '4' for encapsulated IPv4 packets [RFC2003], '41' for encapsulated IPv6 packets [RFC2473][RFC4213], '47' for GRE [RFC1701], '80' for encapsulated OSI/CLNP packets

[RFC1070], etc.

Next, if the inner packet is no larger than (MINMTU-HLEN) or larger than 1500, the ITE sets (M=0; Fragment Offset=0). Otherwise, the ITE breaks the inner packet into N non-overlapping segments, where N is minimized. For IPv6 as the inner protocol, the resulting encapsulated SEAL packet containing the first segment MUST be at least as large as the IPv6 minimum of 1280 bytes so that the entire IPv6 header chain is likely to fit within the first segment. (Since the Fragment Offset field indicates the number of 8 byte units, however, if HLEN is not an integer multiple of 8 bytes the encapsulated SEAL packet MAY contain up to 7 bytes less than 1280 so that the IPv6 minimum MTU is not exceeded.)

The ITE then appends a clone of the SEAL header from the first segment onto the head of each additional segment. The ITE then sets (M=1; Fragment Offset=0) in the first segment, sets (M=0/1; Fragment Offset=O(1)) in the second segment, sets (M=0/1; Fragment Offset=O(2)) in the third segment (if needed), etc., then finally sets (M=0; Fragment Offset=O(n)) in the final segment (where O(i) is the number of 256 byte blocks that preceded this segment).

The ITE then writes a monotonically-incrementing integer value for this ETE in the Identification field beginning with a randomly-initialized value in the first packet transmitted. (For SEAL packets that have been split into multiple pieces, the ITE writes the same Identification value in each piece.) The monotonically-incrementing requirement is to satisfy ETEs that use this value for anti-replay purposes. The value is incremented modulo 2^32, i.e., it wraps back to 0 when the previous value was (2^32 - 1).

When USE_ICV is FALSE, the ITE next sets I=0. Otherwise, the ITE sets I=1, includes a trailing ICV and calculates the MAC using HMAC-SHA-1 with a 160 bit secret key and 80 bit MAC field. Beginning with the SEAL header, the ITE calculates the MAC over the leading 128 bytes of the packet (or up to the end of the packet if there are fewer than 128 bytes) and places the result in the MAC field. (For SEAL packets that have been split into multiple pieces, each piece calculates its own MAC.) The ITE then writes the value 0 in the F flag and 0x00 in the Algorithm field of the ICV control octet (other values for these fields, and other MAC calculation disciplines, are outside the scope of this document and may be specified in future documents.)

If the packet is undergoing SEAL re-encapsulation, the ITE then copies the R value from the SEAL header of the packet to be re-encapsulated. Otherwise, it sets R=0 unless otherwise specified in other documents that employ SEAL. The ITE then adds the outer

encapsulating headers as specified in Section 5.4.5.

5.4.5. Outer Encapsulation

Following SEAL encapsulation, the ITE next encapsulates each segment in the requisite outer transport (when necessary) and IP layer headers. When a transport layer header such as UDP or TCP is included, the ITE writes the port number for SEAL in the transport destination service port field.

When UDP encapsulation is used, the ITE sets the UDP checksum field to zero for IPv4 packets and also sets the UDP checksum field to zero for IPv6 packets even though IPv6 generally requires UDP checksums. Further considerations for setting the UDP checksum field for IPv6 packets are discussed in [RFC6935][RFC6936].

The ITE then sets the outer IP layer headers the same as specified for ordinary IP encapsulation (e.g., [RFC1070][RFC2003], [RFC2473], [RFC4213], etc.) except that for ordinary SEAL packets the ITE copies the "TTL/Hop Limit", "Type of Service/Traffic Class" and "Congestion Experienced" values in the inner network layer header into the corresponding fields in the outer IP header. For transitional SEAL packets undergoing re-encapsulation, the ITE instead copies the "TTL/Hop Limit", "Type of Service/Traffic Class" and "Congestion Experienced" values in the original outer IP header of the transitional packet into the corresponding fields in the new outer IP header of the packet to be forwarded (i.e., the values are transferred between outer headers and *not* copied from the inner network layer header).

The ITE also sets the IP protocol number to the appropriate value for the first protocol layer within the encapsulation (e.g., UDP, TCP, SEAL, etc.). When IPv6 is used as the outer IP protocol, the ITE then sets the flow label value in the outer IPv6 header the same as described in [RFC6438]. When IPv4 is used as the outer IP protocol, the ITE sets DF=0 in the IPv4 header to allow the packet to be fragmented if it encounters a restricting link (for IPv6 SEAL paths, the DF bit is absent but implicitly set to 1).

The ITE finally sends each outer packet via the underlying link corresponding to LINK.

5.4.6. Path Probing and ETE Reachability Verification

All SEAL data packets sent by the ITE are considered implicit probes that detect MTU limitations on the SEAL path, while explicit probe packets can be constructed to probe the path MTU and/or verify ETE reachability. These probes will elicit an SCMP message from the ETE

if it needs to send an acknowledgement and/or report an error condition. The probe packets may also be dropped by either the ETE or a router on the path, which may or may not result in an ICMP message being returned to the ITE.

To generate an explicit probe packet, the ITE creates a duplicate of an actual data packet and uses the duplicate as a probe. (Alternatively, the ITE can create a packet buffer beginning with the same outer headers, SEAL header and inner network layer headers that would appear in an ordinary data packet, then pad the packet with random data.) The ITE then sets (C=0; P=1) in the SEAL header of the probe packet, and also sets DF=1 in the outer IP header when IPv4 is used.

The ITE sends periodic explicit probes to determine whether SEAL segmentation is still necessary (see Section 5.4.4). In particular, if a probe packet of 1500 bytes (i.e., a packet that becomes (1500+ HLEN) bytes after encapsulation) succeeds without incurring fragmentation the ITE is assured that the path MTU is large enough so that the segmentation/reassembly process can be suspended. This probing discipline can therefore be considered as Packetization Layer Path MTU Discovery (PLPMTUD) [RFC4821] applied to tunnels, which operates independently of any application of PLPMTUD between end systems. Note that the explicit probe size of 1500 bytes is chosen since probe packets smaller than this size may be fragmented by a nested ITE further down the path. For example, a successful probe for a packet size of 1400 bytes does not guarantee that fragmentation is not occurring at another ITE.

The ITE can also send probes to detect whether an outer transport layer header is no longer necessary to reach this ETE. For example, if the ITE sends its initial packets as IP/UDP/SEAL/*, it can send probes constructed as IP/SEAL/* to determine whether the ETE is reachable without the added layer of encapsulation. If so, the ITE should also re-probe the path MTU since switching to a new encapsulation type may result in a path change.

While probing, the ITE processes ICMP messages as specified in Section 5.4.7 and processes SCMP messages as specified in Section 5.6.2.

5.4.7. Processing ICMP Messages

When the ITE sends SEAL packets, it may receive ICMP error messages [RFC0792][RFC4443] from a router on the path to the ETE. Each ICMP message includes an outer IP header, followed by an ICMP header, followed by a portion of the SEAL data packet that generated the error (also known as the "packet-in-error"). Note that the ITE may

receive an ICMP message from another ITE that is at the head end of a nested level of encapsulation. The ITE has no security associations with this nested ITE, hence it should consider the message the same as if it originated from an ordinary router on the path to the ETE.

The ITE should process ICMPv4 Protocol Unreachable messages and ICMPv6 Parameter Problem messages with Code "Unrecognized Next Header type encountered" as a hint that the ETE does not implement SEAL. The ITE can optionally ignore other ICMP messages that do not include sufficient information in the packet-in-error, or process them as a hint that the SEAL path to the ETE may be failing. The ITE then discards these types of messages.

For other ICMP messages, the ITE first examines the SEAL data packet within the packet-in-error field. If the IP source and/or destination addresses are invalid, or if the value in the SEAL header Identification field (if present) is not within the window of packets the ITE has recently sent to this ETE, or if the MAC value in the ICV field (if present) is incorrect, the ITE discards the message.

Next, if the received ICMP message is a PTB the ITE sets the temporary variable "PMTU" for this SEAL path to the MTU value in the PTB message. If the outer IP length value in the packet-in-error is no larger than (1500+HLEN) bytes the ITE sets MAXMTU=(1500+HLEN) and discards the message. If the outer IP length value in the packet-in-error is larger than (1500+HLEN) bytes and PMTU is no smaller than MINMTU the ITE sets MAXMTU to the maximum of (1500+HLEN) and PMTU; otherwise the ITE consults a plateau table (e.g., as described in [RFC1191]) to determine a new value for MAXMTU. For example, if the ITE receives a PTB message with small PMTU and packet-in-error length 8KB, it can set MAXMTU=4KB. If the ITE subsequently receives a PTB message with small PMTU and length 4KB, it can set MAXMTU=2KB, etc., to a minimum value of MAXMTU=(1500+HLEN). Next, if the packet-in-error was an explicit probe (i.e., one with P=1 in the SEAL header), the ITE discards the message. Finally, if the ITE is using a MINMTU value larger than 1280 for IPv6 or 576 for IPv4, it may need to reduce MINMTU if the PMTU value is small.

If the ICMP message was not discarded, the ITE transcribes it into a message appropriate for the SEAL data packet within the packet-in-error. If the previous hop toward the inner source address within the SEAL data packet is reached via the same SEAL tunnel, the ITE transcribes the message into an SCMP message the same as described for ETE generation of SCMP messages in Section 5.6.1, i.e., it copies the SEAL data packet within the packet-in-error into the packet-in-error field of the new message. (In this process, the ETE also sets (C=1; P=1) in the SEAL header of the SCMP message.) Otherwise, the ITE seeks beyond the SEAL header within the packet-in-error and

transcribes the inner packet into a message appropriate for the inner protocol version (e.g., ICMPv4 for IPv4, ICMPv6 for IPv6, etc.).

The ITE finally forwards the transcribed message to the previous hop toward the inner source address.

5.4.8. IPv4 Middlebox Reassembly Testing

The ITE can perform a qualification exchange to ensure that the subnetwork correctly delivers fragments to the ETE. This procedure can be used, e.g., to determine whether there are middleboxes on the path that violate the [RFC1812], Section 5.2.6 requirement that: "A router MUST NOT reassemble any datagram before forwarding it". Examples of middleboxes that may perform reassembly include stateful NATs and firewalls. Such devices could still allow for stateless MTU determination if they gather the fragments of a fragmented SEAL data packet for packet analysis purposes but then forward the fragments on to the final destination rather than forwarding the reassembled packet. (This process is often referred to as "Virtual Fragmentation Reassembly" (VFR)).

The ITE should use knowledge of its topological arrangement as an aid in determining when middlebox reassembly testing is necessary. For example, if the ITE is aware that the ETE is located somewhere in the public Internet, middlebox reassembly testing should not be necessary. If the ITE is aware that the ETE is located behind a NAT or a firewall, however, then reassembly testing can be used to detect middleboxes that do not conform to specifications.

The ITE can perform a middlebox reassembly test by sending explicit probe packets. The ITE should only send probe packets that are smaller than (576-HLEN) before encapsulation since the least an ordinary node can be expected to reassemble is 576 bytes. To generate a probe, the ITE either creates a clone of an ordinary data packet or creates a packet buffer beginning with the same outer headers, SEAL header and inner network layer header that would appear in an ordinary data packet. The ITE then pads the probe packet with random data to a length that is at least 128 bytes but smaller than (576-HLEN) bytes.

The ITE then sets (C=0; P=1) in the SEAL header of the probe packet and sets the Next Header field to the inner network layer protocol type. Next, the ITE sets LINK to the appropriate value for this SEAL path, sets the Identification field, then finally calculates the ICV and sets I=1 (when USE_ICV is TRUE).

The ITE then encapsulates the probe packet in the appropriate outer headers, splits it into two outer IP fragments, then sends both

fragments over the same SEAL path.

The ITE should send a series of probe packets (e.g., 3-5 probes with 1sec intervals between tests) instead of a single isolated probe in case of packet loss. If the ETE returns an SCMP PTB message with the original first fragment in the packet-in-error, then the SEAL path correctly supports fragmentation; otherwise, the ITE enables stateful MTU determination for this SEAL path as specified in Section 5.4.9.

5.4.9. Stateful MTU Determination

SEAL supports a stateless MTU determination capability, however the ITE may in some instances wish to impose a stateful MTU limit on a particular SEAL path. For example, when the ETE is situated behind a middlebox that performs reassembly in violation of the specs (see: Section 5.4.8) it is imperative that fragmentation be avoided. In other instances (e.g., when the SEAL path includes performance-constrained links), the ITE may deem it necessary to cache a conservative static MTU in order to avoid sending large packets that would only be dropped due to an MTU restriction somewhere on the path.

To determine a static MTU value, the ITE can send a series of probe packets of various sizes to the ETE with (C=0; P=1) in the SEAL header and DF=1 in the outer IP header. The ITE then caches the size 'S' of the largest packet for which it receives a probe reply from the ETE by setting MAXMTU=MAX((S, (1500+HLEN)) for this SEAL path.

For example, the ITE could send probe packets of 8KB, followed by 4KB, followed by 2KB, etc. While probing, the ITE processes any ICMP PTB message it receives as a potential indication of probe failure then discards the message.

5.4.10. Detecting Path MTU Changes

When stateful MTU determination is used, the ITE SHOULD periodically reset MAXMTU and/or re-probe the path to determine whether MAXMTU has increased. If the path still has a too-small MTU, the ITE will receive a PTB message that reports a smaller size.

5.5. ETE Specification

5.5.1. Reassembly Buffer Requirements

For IPv6, the ETE MUST configure a minimum reassembly buffer size of (1500 + HLEN) bytes for the reassembly of outer IPv6 packets, i.e., even though the true minimum reassembly size for IPv6 is only 1500 bytes [RFC2460]. For IPv4, the ETE also MUST configure a minimum

reassembly buffer size of (1500 + HLEN) bytes for the reassembly of outer IPv4 packets, i.e., even though the true minimum reassembly size for IPv4 is only 576 bytes [RFC1122].

In addition to this outer reassembly buffer requirement, the ETE further MUST configure a minimum SEAL reassembly buffer size of (1500 + HLEN) bytes for the reassembly of segmented SEAL packets (see: Section 5.5.4).

Note that the value "HLEN" may be variable and initially unknown to the ETE, and would typically range from a few bytes to a few tens of bytes or even more. It is therefore RECOMMENDED that the ETE configure slightly larger minimum IP/SEAL reassembly buffer sizes of 2048 bytes (2KB).

5.5.2. Tunnel Neighbor Soft State

When message origin authentication and integrity checking is required, the ETE maintains a per-ITE MAC calculation algorithm and a symmetric secret key to verify the MAC. The ETE also maintains a window of Identification values for the packets it has recently received from this ITE as well as a window of Identification values for the packets it has recently sent to this ITE.

When the tunnel neighbor relationship is bidirectional, the ETE further maintains a per SEAL path mapping of outer IP and transport layer addresses to the LINK value that appears in packets received from the ITE.

5.5.3. IP-Layer Reassembly

The ETE reassembles fragmented IP packets that are explicitly addressed to itself. For IP fragments that are received via a SEAL tunnel, the ETE SHOULD maintain conservative reassembly cache high- and low-water marks. When the size of the reassembly cache exceeds this high-water mark, the ETE SHOULD actively discard stale incomplete reassemblies (e.g., using an Active Queue Management (AQM) strategy) until the size falls below the low-water mark. The ETE SHOULD also actively discard any pending reassemblies that clearly have no opportunity for completion, e.g., when a considerable number of new fragments have arrived before a fragment that completes a pending reassembly arrives.

The ETE processes non-SEAL IP packets as specified in the normative references, i.e., it performs any necessary IP reassembly then discards the packet if it is larger than the reassembly buffer size or delivers the (fully-reassembled) packet to the appropriate upper layer protocol module.

For SEAL packets, the ETE performs any necessary IP reassembly then submits the packet for SEAL decapsulation as specified in Section 5.5.4. (Note that if the packet is larger than the reassembly buffer size, the ETE still examines the leading portion of the (partially) reassembled packet during decapsulation.)

5.5.4. Decapsulation, SEAL-Layer Reassembly, and Re-Encapsulation

For each SEAL packet accepted for decapsulation, the ETE first examines the Identification field. If the Identification is not within the window of acceptable values for this ITE, the ETE silently discards the packet.

Next, if I==1 the ETE SHOULD verify the MAC value and silently discard the packet if the value is incorrect. (Note that this means that the ETE would need to receive all IP fragments if the packet was fragmented at the outer IP layer, since the MAC is included as a trailing field.)

Next, if the packet arrived as multiple IP fragments, the ETE sends an SPTB message back to the ITE with MTU set to the size of the largest fragment received (see: Section 5.6.1.1).

Next, if the packet arrived as multiple IP fragments and the inner packet is larger than 1500 bytes, the ETE silently discards the packet; otherwise, it continues to process the packet.

Next, if there is an incorrect value in a SEAL header field (e.g., an incorrect "VER" field value), the ETE discards the packet. If the SEAL header has C==0, the ETE also returns an SCMP "Parameter Problem" (SPP) message (see Section 5.6.1.2).

Next, if the SEAL header has C==1, the ETE processes the packet as an SCMP packet as specified in Section 5.6.2. Otherwise, the ETE continues to process the packet as a SEAL data packet.

Next, if the SEAL header has (M==1 || Fragment Offset!=0) the ETE checks to see if the other segments of this already-segmented SEAL packet have arrived, i.e., by looking for additional segments that have the same outer IP source address, destination address, source port number and SEAL Identification value. If all other segments have already arrived, the ETE discards the SEAL header and other outer headers from the non-initial segments and appends the segments onto the end of the first segment according to their offset value. Otherwise, the ETE caches the new segment for at most 60 seconds while awaiting the arrival of its partners. During this process, the ETE discards any segments that are overlapping with respect to segments that have already been received, and also discards any

segments that have M==1 in the SEAL header but do not contain an integer multiple of 8 bytes. The ETE further SHOULD manage the SEAL reassembly cache the same as described for the IP-Layer Reassembly cache in Section 5.5.3, i.e., it SHOULD perform an early discard for any pending reassemblies that have low probability of completion.

Next, if the SEAL header in the (reassembled) packet has P==1, the ETE drops the packet unconditionally and sends an SPTB message back to the ITE (see: Section 5.6.1.1) if it has not already sent an SPTB message based on IP fragmentation. (Note that the ETE therefore sends only a single SPTB message for a probe packet that also experienced IP fragmentation, i.e., it does not send multiple SPTB messages.)

Finally, the ETE discards the outer headers and processes the inner packet according to the header type indicated in the SEAL Next Header field. If the next hop toward the inner destination address is via a different interface than the SEAL packet arrived on, the ETE discards the SEAL header and delivers the inner packet either to the local host or to the next hop if the packet is not destined to the local host.

If the next hop is on the same tunnel the SEAL packet arrived on, however, the ETE submits the packet for SEAL re-encapsulation beginning with the specification in Section 5.4.3 above and without decrementing the value in the inner (TTL / Hop Limit) field.

5.6. The SEAL Control Message Protocol (SCMP)

SEAL provides a companion SEAL Control Message Protocol (SCMP) that uses the same message types and formats as for the Internet Control Message Protocol for IPv6 (ICMPv6) [RFC4443]. The SCMP messaging protocol operates over bidirectional and partially unidirectional tunnels. (For fully unidirectional tunnels, SEAL must operate without the benefit of SCMP meaning that steady-state fragmentation and reassembly may be necessary in extreme cases. In that case, the ITE must select a conservative MINMTU to ensure that IPv4 fragmentation is avoided in order to avoid reassembly errors at high data rates [RFC4963].)

As for ICMPv6, each SCMP message includes a 32-bit header and a variable-length body. The ITE encapsulates the SCMP message in a SEAL header and outer headers as shown in Figure 4:

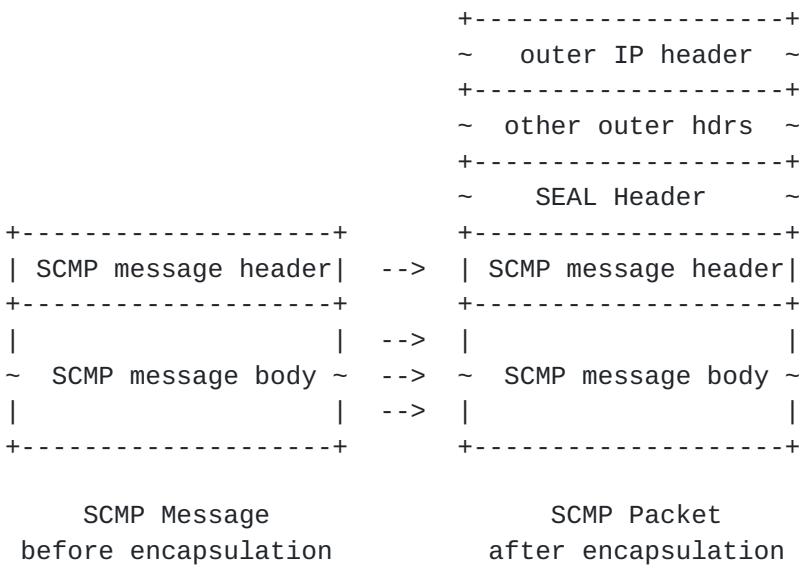
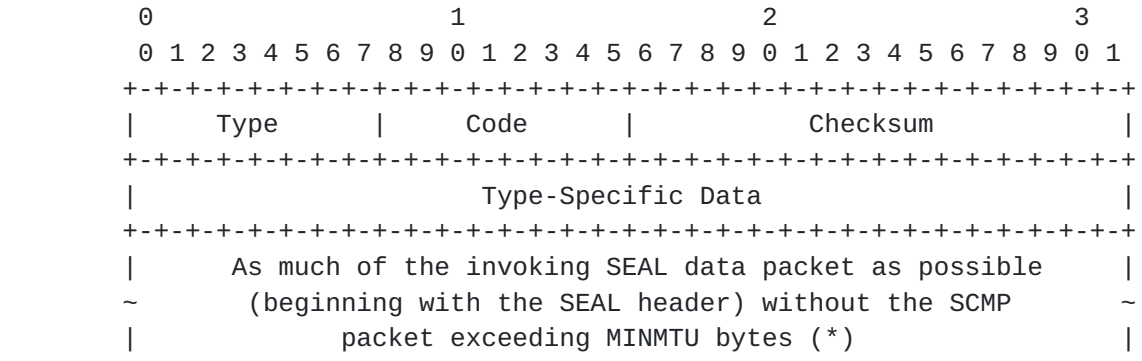


Figure 4: SCMP Message Encapsulation

The following sections specify the generation, processing and relaying of SCMP messages.

5.6.1. Generating SCMP Messages

ETEs generate SCMP messages in response to receiving certain SEAL data packets using the format shown in Figure 5:



(*) also known as the "packet-in-error"

Figure 5: SCMP Message Format

The error message includes the 32-bit SCMP message header, followed by a 32-bit Type-Specific Data field, followed by the leading portion of the invoking SEAL data packet beginning with the SEAL header as the "packet-in-error". The packet-in-error includes as much of the invoking packet as possible extending to a length that would not cause the entire SCMP packet following outer encapsulation to exceed

MINMTU bytes.

When the ETE processes a SEAL data packet for which the Identification and ICV values are correct but an error must be returned, it prepares an SCMP message as shown in Figure 5. The ETE sets the Type and Code fields to the same values that would appear in the corresponding ICMPv6 message [RFC4443], but calculates the Checksum beginning with the SCMP message header using the algorithm specified for ICMPv4 in [RFC0792].

The ETE next encapsulates the SCMP message in the requisite SEAL and outer headers as shown in Figure 4. During encapsulation, the ETE sets the outer destination address/port numbers of the SCMP packet to the values associated with the ITE and sets the outer source address/port numbers to its own outer address/port numbers.

The ETE then sets (C=1; M=0; Fragment Offset=0) in the SEAL header, then sets I, Next Header and LINK to the same values that appeared in the SEAL header of the data packet. The ETE next sets the Identification field to the next Identification value scheduled for this ITE, then increments the next Identification value. When I==1, the ETE then prepares the ICV field the same as specified for SEAL data packet encapsulation in Section 5.4.4. If this message is in direct response to a SEAL data packet sent by the ITE, the ETE next sets P=0 and sends the resulting SCMP packet to the ITE the same as specified for SEAL data packets in Section 5.4.5.

If the message is in response to an SCMP message received from a next hop ETE or to an ICMP message received from a router on the path to a next hop ETE, the ETE instead sets P=1 and passes the message to the ITE in a "reverse re-encapsulation" process. In particular, when the previous hop toward the source of the inner packet within the packet-in-error in a received SCMP/ICMP message is reached via the same tunnel as the message arrived on, the ETE replaces the outer headers of the message (up to and including the SEAL header) with headers that will be recognized and accepted by the previous hop and sends the resulting packet to the previous hop.

The following sections describe additional considerations for various SCMP error messages:

5.6.1.1. Generating SCMP Packet Too Big (SPTB) Messages

An ETE generates an SPTB message when it receives a SEAL probe packet (i.e., one with C=0; P=1 in the SEAL header) or when it receives a SEAL packet that arrived as multiple outer IP fragments. The ETE prepares the SPTB message the same as for the corresponding ICMPv6 PTB message, and writes the length of the largest outer IP fragment

received in the MTU field of the message (or the full length of the outer IP packet if the packet was unfragmented). In that case, the ETE sets (C=1; P=0) in the SEAL header.

An ETE also generates an SPTB message when it attempts to forward a SEAL data packet to a next hop ETE via the same tunnel the data packet arrived on, but for which MAXMTU for that SEAL path is insufficient to accommodate the packet (See Section 5.4.3.2). In that case, the ETE sets (C=1; P=1) in the SEAL header.

An ETE finally generates an SPTB message when it receives an ICMP PTB message from a router on the path to a next hop ETE (See Section 5.4.7). In that case, the ETE also sets (C=1; P=1) in the SEAL header.

5.6.1.2. Generating Other SCMP Messages

An ETE generates an SCMP "Destination Unreachable" (SDU) message under the same conditions that an IPv6 system would generate an ICMPv6 Destination Unreachable message.

An ETE generates an SCMP "Parameter Problem" (SPP) message when it receives a SEAL packet with an incorrect value in the SEAL header.

TEs generate other SCMP message types using methods and procedures specified in other documents. For example, SCMP message types used for tunnel neighbor coordinations are specified in VET [I-D.templin-intarea-vet].

5.6.2. Processing SCMP Messages

An ITE may receive SCMP messages with C==1 in the SEAL header after sending packets to an ETE. The ITE first verifies that the outer addresses of the SCMP packet are correct, and that the Identification field contains an acceptable value. The ITE next verifies that the SEAL header fields are set correctly as specified in Section 5.6.1. When I==1, the ITE then verifies the ICV. The ITE next verifies the Checksum value in the SCMP message header. If any of these values are incorrect, the ITE silently discards the message; otherwise, it processes the message as follows:

5.6.2.1. Processing SCMP PTB Messages

After an ITE sends a SEAL packet to an ETE, it may receive an SPTB message with a packet-in-error containing the leading portion of the packet (see: Section 5.6.1.1). If the SEAL header has P==1 the ITE consults its forwarding information base to pass the message to the previous hop toward the source address of the encapsulated inner

packet. When the previous hop is reached via the same SEAL tunnel, the ITE passes the SPTB message to the previous hop as specified in Section 5.6.1. Otherwise, the ITE transcribes the inner packet within the packet-in-error into a message appropriate for the inner protocol version (e.g., ICMPv4 for IPv4, ICMPv6 for IPv6, etc.).

If the SEAL header has P==0, the ITE instead processes the message as an MTU limitation on the SEAL path to this ETE. In that case, the ITE first sets the temporary variable "PMTU" for this SEAL path to the MTU value in the SPTB message and processes the message as follows:

- If PMTU is no smaller than (1500+HLEN), the ITE suspends the SEAL segmentation/reassembly process for this SEAL path so that whole (unfragmented) SEAL packets can be used. If the packet is a probe being used to establish a stateful MTU for this SEAL path (see: section 5.4.9), the ITE also sets MAXMTU=PMTU.

- If PMTU is smaller than (1500+HLEN) but no smaller than MINMTU the ITE sets MAXMTU to (1500+HLEN) and resumes the SEAL segmentation/ reassembly process for this SEAL path.

- If PMTU is smaller than MINMTU and the packet-in-error is a probe used for the purpose of middlebox reassembly detection (see: section 5.4.8), the ITE notes the results of the probe. Otherwise, the ITE consults a plateau table to determine a new value for MAXMTU. For example, if the ITE receives a PTB message with small PMTU and packet-in-error length 8KB, it can set MAXMTU=4KB. If the ITE subsequently receives a PTB message with small PMTU and length 4KB, it can set MAXMTU=2KB, etc., to a minimum value of MAXMTU=(1500+HLEN). Finally, if the ITE is using a MINMTU value larger than 1280 for IPv6 or 576 for IPv4, it may need to reduce MINMTU if the PMTU value is small.

Next, if the packet-in-error was no larger than (1500+HLEN) or the packet-in-error was an explicit probe (i.e., one with (C==0; P==1 in the SEAL header of the packet-in-error), the ITE discards the SPTB message.

5.6.2.2. Processing Other SCMP Error Messages

An ITE may receive an SDU message with an appropriate code under the same circumstances that an IPv6 node would receive an ICMPv6 Destination Unreachable message. The ITE transcribes the message and forwards it toward the source address of the inner packet within the packet-in-error the same as specified for SPTB messages with P==1 in Section 5.6.2.1.

An ITE may receive an SPP message when the ETE receives a SEAL packet with an incorrect value in the SEAL header. The ITE should examine the SEAL header within the packet-in-error to determine whether different settings should be used in subsequent packets, but does not relay the message further.

TEs process other SCMP message types using methods and procedures specified in other documents. For example, SCMP message types used for tunnel neighbor coordinations are specified in VET [I-D.templin-intarea-vet].

6. Link Requirements

Subnetwork designers are expected to follow the recommendations in Section 2 of [RFC3819] when configuring link MTUs.

7. End System Requirements

End systems are encouraged to implement end-to-end MTU assurance (e.g., using Packetization Layer Path MTU Discovery (PLPMTUD) per [RFC4821]) even if the subnetwork is using SEAL.

When end systems use PLPMTUD, SEAL will ensure that the tunnel behaves as a link in the path that assures an MTU of at least 1500 bytes while not precluding discovery of larger MTUs. The PLPMTUD mechanism will therefore be able to function as designed in order to discover and utilize larger MTUs.

8. Router Requirements

Routers within the subnetwork are expected to observe the standard IP router requirements, including the implementation of IP fragmentation and reassembly as well as the generation of ICMP messages [RFC0792][RFC1122][RFC1812][RFC2460][RFC4443][RFC6434].

Note that, even when routers support existing requirements for the generation of ICMP messages, these messages are often filtered and discarded by middleboxes on the path to the original source of the message that triggered the ICMP. It is therefore not possible to assume delivery of ICMP messages even when routers are correctly implemented.

9. Nested Encapsulation Considerations

SEAL supports nested tunneling - an example would be a recursive nesting of mobile networks, where the first network receives service from an ISP, the second network receives service from the first network, the third network receives service from the second network, etc. It is imperative that such nesting not extend indefinitely; SEAL tunnels therefore honor the Encapsulation Limit option defined in [RFC2473].

In such nested arrangements, the SEAL ITE has a tunnel neighbor relationship only with ETEs at its own nesting level, i.e., it does not have a tunnel neighbor relationship with TEs at other nesting levels.Therefore, when an ITE 'A' within an outer nesting level needs to return an error message to an ITE 'B' within an inner nesting level, it generates an ordinary ICMP error message the same as if it were an ordinary router within the subnetwork. 'B' can then perform message validation as specified in Section 5.4.7, but full message origin authentication is not possible.

(Note that the SCMP protocol could instead be extended to allow an outer nesting level ITE 'A' to return an SCMP message to an inner nesting level ITE 'B' rather than return an ICMP message. This would conceptually allow the control messages to pass through firewalls and NATs, however it would give no more message origin authentication assurance than for ordinary ICMP messages. It was therefore determined that the complexity of extending the SCMP protocol was of little value within the context of the anticipated use cases for nested encapsulations.)

10. Reliability Considerations

Although a SEAL tunnel may span an arbitrarily-large subnetwork expanse, the IP layer sees the tunnel as a simple link that supports the IP service model. Links with high bit error rates (BERs) (e.g., IEEE 802.11) use Automatic Repeat-ReQuest (ARQ) mechanisms [RFC3366] to increase packet delivery ratios, while links with much lower BERs typically omit such mechanisms. Since SEAL tunnels may traverse arbitrarily-long paths over links of various types that are already either performing or omitting ARQ as appropriate, it would therefore be inefficient to require the tunnel endpoints to also perform ARQ.

11. Integrity Considerations

The SEAL header includes an integrity check field that covers the SEAL header and at least the inner packet headers. This provides for

header integrity verification on a segment-by-segment basis for a segmented re-encapsulating tunnel path.

Fragmentation and reassembly schemes must also consider packet-splicing errors, e.g., when two fragments from the same packet are concatenated incorrectly, when a fragment from packet X is reassembled with fragments from packet Y, etc. The primary sources of such errors include implementation bugs and wrapping IPv4 ID fields.

In particular, the IPv4 16-bit ID field can wrap with only 64K packets with the same (src, dst, protocol)-tuple alive in the system at a given time [RFC4963]. When the IPv4 ID field is re-written by a middlebox such as a NAT or Firewall, ID field wrapping can occur with even fewer packets alive in the system. It is therefore essential that IPv4 fragmentation and reassembly be detected early and tuned out through proper application of SEAL segmentation and reassembly.

12. IANA Considerations

The IANA is requested to allocate a User Port number for "SEAL" in the 'port-numbers' registry. The Service Name is "SEAL", and the Transport Protocols are TCP and UDP. The Assignee is the IESG (iesg@ietf.org) and the Contact is the IETF Chair (chair@ietf.org). The Description is "Subnetwork Encapsulation and Adaptation Layer (SEAL)", and the Reference is the RFC-to-be currently known as 'draft-templin-intarea-seal'.

13. Security Considerations

SEAL provides a segment-by-segment message origin authentication, integrity and anti-replay service. The SEAL header is sent in-the-clear the same as for the outer IP and other outer headers. In this respect, the threat model is no different than for IPv6 extension headers. Unlike IPv6 extension headers, however, the SEAL header can be protected by an integrity check that also covers the inner packet headers.

An amplification/reflection/buffer overflow attack is possible when an attacker sends IP fragments with spoofed source addresses to an ETE in an attempt to clog the ETE's reassembly buffer and/or cause the ETE to generate a stream of SCMP messages returned to a victim ITE. The SCMP message ICV, Identification, as well as the inner headers of the packet-in-error, provide mitigation for the ETE to detect and discard SEAL segments with spoofed source addresses.

Security issues that apply to tunneling in general are discussed in [RFC6169].

14. Related Work

Section 3.1.7 of [RFC2764] provides a high-level sketch for supporting large tunnel MTUs via a tunnel-level segmentation and reassembly capability to avoid IP level fragmentation.

Section 3 of [RFC4459] describes inner and outer fragmentation at the tunnel endpoints as alternatives for accommodating the tunnel MTU.

Section 4 of [RFC2460] specifies a method for inserting and processing extension headers between the base IPv6 header and transport layer protocol data. The SEAL header is inserted and processed in exactly the same manner.

IPsec/AH is [RFC4301][RFC4301] is used for full message integrity verification between tunnel endpoints, whereas SEAL only ensures integrity for the inner packet headers. The AYIYA proposal [I-D.massar-v6ops-ayiya] uses similar means for providing message authentication and integrity.

SEAL, along with the Virtual Enterprise Traversal (VET) [I-D.templin-intarea-vet] tunnel virtual interface abstraction, are the functional building blocks for the Interior Routing Overlay Network (IRON) [I-D.templin-ironbis] and Routing and Addressing in Networks with Global Enterprise Recursion (RANGER) [RFC5720][RFC6139] architectures.

The concepts of path MTU determination through the report of fragmentation and extending the IPv4 Identification field were first proposed in deliberations of the TCP-IP mailing list and the Path MTU Discovery Working Group (MTUDWG) during the late 1980's and early 1990's. An historical analysis of the evolution of these concepts, as well as the development of the eventual PMTUD mechanism, appears in [RFC5320].

15. Implementation Status

An early implementation of the first revision of SEAL [RFC5320] is available at: http://isatap.com/seal.

16. Acknowledgments

The following individuals are acknowledged for helpful comments and suggestions: Jari Arkko, Fred Baker, Iljitsch van Beijnum, Oliver Bonaventure, Teco Boot, Bob Braden, Brian Carpenter, Steve Casner, Ian Chakeres, Noel Chiappa, Remi Denis-Courmont, Remi Despres, Ralph Droms, Aurnaud Ebalard, Gorry Fairhurst, Washam Fan, Dino Farinacci, Joel Halpern, Brian Haberman, Sam Hartman, John Heffner, Thomas Henderson, Bob Hinden, Christian Huitema, Eliot Lear, Darrel Lewis, Joe Macker, Matt Mathis, Erik Nordmark, Dan Romascanu, Dave Thaler, Joe Touch, Mark Townsley, Ole Troan, Margaret Wasserman, Magnus Westerlund, Robin Whittle, James Woodyatt, and members of the Boeing Research & Technology NST DC&NT group.

Discussions with colleagues following the publication of [RFC5320] have provided useful insights that have resulted in significant improvements to this, the Second Edition of SEAL.

This document received substantial review input from the IESG and IETF area directorates in the February 2013 timeframe. IESG members and IETF area directorate representatives who contributed helpful comments and suggestions are gratefully acknowledged. Discussions on the IETF IPv6 and Intarea mailing lists in the summer 2013 timeframe also stimulated several useful ideas.

Path MTU determination through the report of fragmentation was first proposed by Charles Lynn on the TCP-IP mailing list in 1987. Extending the IP identification field was first proposed by Steve Deering on the MTUDWG mailing list in 1989. Steve Deering also proposed the IPv6 minimum MTU of 1280 bytes on the IPng mailing list in 1997.

17. References

17.1. Normative References

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.

[RFC0792] Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, September 1981.

[RFC1122] Braden, R., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, October 1989.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[RFC3971] Arkko, J., Kempf, J., Zill, B., and P. Nikander, "SEcure Neighbor Discovery (SEND)", RFC 3971, March 2005.

[RFC4443] Conta, A., Deering, S., and M. Gupta, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 4443, March 2006.

[RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.

17.2. Informative References

[FOLK] Shannon, C., Moore, D., and k. claffy, "Beyond Folklore: Observations on Fragmented Traffic", December 2002.

[FRAG] Kent, C. and J. Mogul, "Fragmentation Considered Harmful", October 1987.

[I-D.ietf-6man-ext-transmit] Carpenter, B. and S. Jiang, "Transmission and Processing of IPv6 Extension Headers", draft-ietf-6man-ext-transmit-05 (work in progress), October 2013.

[I-D.massar-v6ops-ayiya] Massar, J., "AYIYA: Anything In Anything", draft-massar-v6ops-ayiya-02 (work in progress), July 2004.

[I-D.taylor-v6ops-fragdrop] Jaeggli, J., Colitti, L., Kumari, W., Vyncke, E., Kaeo, M., and T. Taylor, "Why Operators Filter Fragments and What It Implies", draft-taylor-v6ops-fragdrop-01 (work in progress), June 2013.

[I-D.templin-intarea-vet] Templin, F., "Virtual Enterprise Traversal (VET)", draft-templin-intarea-vet-40 (work in progress), May 2013.

[I-D.templin-ironbis] Templin, F., "The Interior Routing Overlay Network (IRON)", draft-templin-ironbis-15 (work in progress), May 2013.

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768,

August 1980.

[RFC0994] International Organization for Standardization (ISO) and American National Standards Institute (ANSI), "Final text of DIS 8473, Protocol for Providing the Connectionless-mode Network Service", RFC 994, March 1986.

[RFC1063] Mogul, J., Kent, C., Partridge, C., and K. McCloghrie, "IP MTU discovery options", RFC 1063, July 1988.

[RFC1070] Hagens, R., Hall, N., and M. Rose, "Use of the Internet as a subnetwork for experimentation with the OSI network layer", RFC 1070, February 1989.

[RFC1146] Zweig, J. and C. Partridge, "TCP alternate checksum options", RFC 1146, March 1990.

[RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191, November 1990.

[RFC1701] Hanks, S., Li, T., Farinacci, D., and P. Traina, "Generic Routing Encapsulation (GRE)", RFC 1701, October 1994.

[RFC1812] Baker, F., "Requirements for IP Version 4 Routers", RFC 1812, June 1995.

[RFC1981] McCann, J., Deering, S., and J. Mogul, "Path MTU Discovery for IP version 6", RFC 1981, August 1996.

[RFC2003] Perkins, C., "IP Encapsulation within IP", RFC 2003, October 1996.

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.

[RFC2473] Conta, A. and S. Deering, "Generic Packet Tunneling in IPv6 Specification", RFC 2473, December 1998.

[RFC2675] Borman, D., Deering, S., and R. Hinden, "IPv6 Jumbograms", RFC 2675, August 1999.

[RFC2764] Gleeson, B., Heinanen, J., Lin, A., Armitage, G., and A. Malis, "A Framework for IP Based Virtual Private Networks", RFC 2764, February 2000.

[RFC2780] Bradner, S. and V. Paxson, "IANA Allocation Guidelines For Values In the Internet Protocol and Related Headers",

BCP 37, RFC 2780, March 2000.

[RFC2827] Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", BCP 38, RFC 2827, May 2000.

[RFC2923] Lahey, K., "TCP Problems with Path MTU Discovery", RFC 2923, September 2000.

[RFC3232] Reynolds, J., "Assigned Numbers: RFC 1700 is Replaced by an On-line Database", RFC 3232, January 2002.

[RFC3366] Fairhurst, G. and L. Wood, "Advice to link designers on link Automatic Repeat reQuest (ARQ)", BCP 62, RFC 3366, August 2002.

[RFC3819] Karn, P., Bormann, C., Fairhurst, G., Grossman, D., Ludwig, R., Mahdavi, J., Montenegro, G., Touch, J., and L. Wood, "Advice for Internet Subnetwork Designers", BCP 89, RFC 3819, July 2004.

[RFC4191] Draves, R. and D. Thaler, "Default Router Preferences and More-Specific Routes", RFC 4191, November 2005.

[RFC4213] Nordmark, E. and R. Gilligan, "Basic Transition Mechanisms for IPv6 Hosts and Routers", RFC 4213, October 2005.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.

[RFC4302] Kent, S., "IP Authentication Header", RFC 4302, December 2005.

[RFC4459] Savola, P., "MTU and Fragmentation Issues with In-the-Network Tunneling", RFC 4459, April 2006.

[RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU Discovery", RFC 4821, March 2007.

[RFC4963] Heffner, J., Mathis, M., and B. Chandler, "IPv4 Reassembly Errors at High Data Rates", RFC 4963, July 2007.

[RFC4987] Eddy, W., "TCP SYN Flooding Attacks and Common Mitigations", RFC 4987, August 2007.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC5320] Templin, F., "The Subnetwork Encapsulation and Adaptation Layer (SEAL)", RFC 5320, February 2010.

[RFC5445] Watson, M., "Basic Forward Error Correction (FEC) Schemes", RFC 5445, March 2009.

[RFC5720] Templin, F., "Routing and Addressing in Networks with Global Enterprise Recursion (RANGER)", RFC 5720, February 2010.

[RFC5927] Gont, F., "ICMP Attacks against TCP", RFC 5927, July 2010.

[RFC6139] Russert, S., Fleischman, E., and F. Templin, "Routing and Addressing in Networks with Global Enterprise Recursion (RANGER) Scenarios", RFC 6139, February 2011.

[RFC6169] Krishnan, S., Thaler, D., and J. Hoagland, "Security Concerns with IP Tunneling", RFC 6169, April 2011.

[RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, August 2011.

[RFC6434] Jankiewicz, E., Loughney, J., and T. Narten, "IPv6 Node Requirements", RFC 6434, December 2011.

[RFC6438] Carpenter, B. and S. Amante, "Using the IPv6 Flow Label for Equal Cost Multipath Routing and Link Aggregation in Tunnels", RFC 6438, November 2011.

[RFC6864] Touch, J., "Updated Specification of the IPv4 ID Field", RFC 6864, February 2013.

[RFC6935] Eubanks, M., Chimento, P., and M. Westerlund, "IPv6 and UDP Checksums for Tunneled Packets", RFC 6935, April 2013.

[RFC6936] Fairhurst, G. and M. Westerlund, "Applicability Statement for the Use of IPv6 UDP Datagrams with Zero Checksums", RFC 6936, April 2013.

[RIPE] De Boer, M. and J. Bosma, "Discovering Path MTU Black Holes on the Internet using RIPE Atlas", July 2012.

[SIGCOMM] Luckie, M. and B. Stasiewicz, "Measuring Path MTU Discovery Behavior", November 2010.

[TBIT] Medina, A., Allman, M., and S. Floyd, "Measuring Interactions Between Transport Protocols and Middleboxes", October 2004.

[WAND] Luckie, M., Cho, K., and B. Owens, "Inferring and Debugging Path MTU Discovery Failures", October 2005.

Author's Address

Fred L. Templin (editor)
Boeing Research & Technology
P.O. Box 3707
Seattle, WA 98124
USA

Email: fltemplin@acm.org