Internet Key Exchange Protocol Version 2 (IKEv2) Message Fragmentation

Abstract

   This document describes a way to avoid IP fragmentation of large
   Internet Key Exchange Protocol version 2 (IKEv2) messages. This
   allows IKEv2 messages to traverse network devices that do not allow
   IP fragments to pass through.

Status of This Memo

   This is an Internet Standards Track document.

   This document is a product of the Internet Engineering Task Force
   (IETF). It represents the consensus of the IETF community. It has
   received public review and has been approved for publication by the
   Internet Engineering Steering Group (IESG). Further information on
   Internet Standards is available in Section 2 of RFC 5741.

   Information about the current status of this document, any errata,
   and how to provide feedback on it may be obtained at
   http://www.rfc-editor.org/info/rfc7383.

Table of Contents

1. Introduction

1.1. Problem Description

   The Internet Key Exchange Protocol version 2 (IKEv2), specified in
   [RFC7296], uses UDP as a transport for its messages. Most IKEv2
   messages are relatively small, usually below several hundred bytes.
   A notable exception is the IKE_AUTH exchange, which requires fairly
   large messages, up to several KB, especially when certificates are
   transferred. When the IKE message size exceeds the path MTU, it gets
   fragmented at the IP level. The problem is that some network
   devices, specifically some NAT boxes, do not allow IP fragments to
   pass through. This apparently blocks IKE communication and,
   therefore, prevents peers from establishing an IPsec Security
   Association (SA). Section 2 of [RFC7296] discusses the impact of IP
   fragmentation on IKEv2 and acknowledges this problem.

   Widespread deployment of Carrier-Grade NATs (CGNs) introduces new
   challenges. [RFC6888] describes requirements for CGNs. It states
   that CGNs must comply with Section 11 of [RFC4787], which requires
   NATs to support receiving IP fragments (REQ-14). In real life,
   fulfillment of this requirement creates an additional burden in terms
   of memory, especially for high-capacity devices used in CGNs. It was
   found by people deploying IKE that more and more ISPs use equipment
   that drops IP fragments, thereby violating this requirement.

   Security researchers have found, and continue to find, attack vectors
   that rely on IP fragmentation. For these reasons, and also as
   articulated in [FRAGDROP], many network operators filter all IPv6
   fragments. Also, the default behavior of many currently deployed
   firewalls is to discard IPv6 fragments.

   In one recent study [BLACKHOLES], two researchers utilized a
   measurement network to measure fragment filtering. They sent
   packets, fragmented to the minimum MTU of 1280, to 502 IPv6-enabled
   and reachable probes. They found that during any given trial period,
   ten percent of the probes did not receive fragmented packets.

   Thus, this problem is valid for both IPv4 and IPv6 and may be caused
   by either deficiency of network devices or operational choice.

1.2. Proposed Solution

   The solution to the problem described in this document is to perform
   fragmentation of large messages by IKEv2 itself and replace them with
   a series of smaller messages. In this case, the resulting IP
   datagrams will be small enough so that no fragmentation at the IP
   level will take place.

   The primary goal of this solution is to allow IKEv2 to operate in
   environments that might block IP fragments. This goal does not
   assume that IP fragmentation should be avoided completely, but only
   in those cases when it interferes with IKE operations. However, this
   solution could be used to avoid IP fragmentation in all situations
   where fragmentation within IKE is applicable, as recommended in
   Section 3.2 of [RFC5405]. Avoiding IP fragmentation would be
   beneficial for IKEv2 in general. The Security Considerations section
   of [RFC7296] mentions exhaustion of the IP reassembly buffers as one
   of the possible attacks on the protocol. In [DOSUDPPROT], several
   aspects of attacks on IKE using IP fragmentation are discussed, and
   one of the defenses it proposes is to perform fragmentation within
   IKE, similar to the solution described in this document.

1.3. Conventions Used in This Document

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

2. Protocol Details

2.1. Overview

   The idea of the protocol described in this document is to split large
   IKEv2 messages into a set of smaller ones, called IKE Fragment
   messages. Fragmentation takes place before the original message is
   encrypted and authenticated, so that each IKE Fragment message
   receives individual protection. On the receiving side, IKE Fragment
   messages are collected, verified, decrypted, and merged together to
   get the original message before encryption. See Appendix A for
   details on design rationale.

2.2. Limitations

   Since IKE Fragment messages are cryptographically protected, SK_a and
   SK_e must already be calculated. In general, it means that the
   original message can be fragmented if and only if it contains an
   Encrypted payload.

   This implies that messages of the IKE_SA_INIT exchange cannot be
   fragmented. In most cases, this is not a problem because IKE_SA_INIT
   messages are usually small enough to avoid IP fragmentation. But in
   some cases (advertising a badly structured long list of algorithms,
   using large Modular Exponentiation (MODP) groups, etc.), these
   messages may become fairly large and get fragmented at the IP level.
   In this case, the solution described in this document will not help.

   Among existing IKEv2 extensions, messages of an IKE_SESSION_RESUME
   exchange, as defined in [RFC5723], cannot be fragmented either. See
   Section 3 for details.

   Another limitation is that the minimum size of an IP datagram bearing
   an IKE Fragment message is about 100 bytes, depending on the
   algorithms employed. According to [RFC0791], the minimum IPv4
   datagram size that is guaranteed not to be further fragmented is
   68 bytes. So, even the smallest IKE Fragment messages could be
   fragmented at the IP level in some circumstances. But such extremely
   small Path MTU (PMTU) sizes are very rare in real life.

2.3. Negotiation

   The initiator indicates its support for IKE fragmentation and
   willingness to use it by including a Notification payload of type
   IKEV2_FRAGMENTATION_SUPPORTED in the IKE_SA_INIT request message. If
   the responder also supports this extension and is willing to use it,
   it includes this notification in the response message.

   Initiator   Responder
   -----------   -----------
   HDR, SAi1, KEi, Ni,
      [N(IKEV2_FRAGMENTATION_SUPPORTED)] -->

                             <--   HDR, SAr1, KEr, Nr, [CERTREQ],
                                   [N(IKEV2_FRAGMENTATION_SUPPORTED)]

   The Notify payload is formatted as follows:

                        1   2   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+
   | Next Payload |C| RESERVED | Payload Length |
   +-+
   |Protocol ID(=0)| SPI Size (=0) | Notify Message Type |
   +-+

   o Protocol ID (1 octet) - MUST be 0.

   o SPI Size (1 octet) - MUST be 0, meaning no Security Parameter
      Index (SPI) is present.

   o Notify Message Type (2 octets) - MUST be 16430, the value assigned
      for the IKEV2_FRAGMENTATION_SUPPORTED notification.

   This notification contains no data.

2.4. Using IKE Fragmentation

   IKE fragmentation MUST NOT be used unless both peers have indicated
   their support for it. After that, it is up to the initiator of each
   exchange to decide whether or not to use it. The responder usually
   replies in the same form as the request message, but other
   considerations might override this.

   The initiator can employ various policies regarding the use of IKE
   fragmentation. It might first try to send an unfragmented message
   and resend it as fragmented only if no complete response is received
   even after several retransmissions. Alternatively, it might choose

to always send fragmented messages (however, see Section 3), or it
might fragment only large messages and messages that are expected to
result in large responses.

The following general guidelines apply:

o If either peer has information that a part of the transaction is
   likely to be fragmented at the IP layer, causing interference with
   the IKE exchange, that peer SHOULD use IKE fragmentation. This
   information might be passed from a lower layer, provided by
   configuration, or derived through heuristics. Examples of
   heuristics are the lack of a complete response after several
   retransmissions for the initiator, and receiving repeated
   retransmissions of the request for the responder.

o If either peer knows that IKE fragmentation has been used in a
   previous exchange in the context of the current IKE SA, that peer
   SHOULD continue to use IKE fragmentation for the messages that are
   larger than the current fragmentation threshold (see
   Section 2.5.1).

o IKE fragmentation SHOULD NOT be used in cases where IP-layer
   fragmentation of both the request and response messages is
   unlikely. For example, there is no point in fragmenting liveness
   check messages.

o If none of the above apply, the responder SHOULD respond in the
   same form (fragmented or not) as the request message to which it
   is responding. Note that the other guidelines might override this
   because of information or heuristics available to the responder.

In most cases, IKE fragmentation will be used in the IKE_AUTH
exchange, especially if certificates are employed.

2.5. Fragmenting Message

Only messages that contain an Encrypted payload are subject to IKE
fragmentation. For the purpose of construction of IKE Fragment
messages, the original (unencrypted) content of the Encrypted payload
is split into chunks. The content is treated as a binary blob and is
split regardless of the boundaries of inner payloads. Each of the
resulting chunks is treated as an original content of the Encrypted
Fragment payload and is then encrypted and authenticated. Thus, the
Encrypted Fragment payload contains a chunk of the original content
of the Encrypted payload in encrypted form. The cryptographic
processing of the Encrypted Fragment payload is identical to that

described in Section 3.14 of [RFC7296], as well as documents updating
such processing for particular algorithms or modes, such as
[RFC5282].

As is the case for the Encrypted payload, the Encrypted Fragment
payload, if present in a message, MUST be the last payload in the
message.

The Encrypted Fragment payload is denoted SKF{...}, and its payload
type is 53. This payload is also called the "Encrypted and
Authenticated Fragment" payload.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Next Payload  |C|  RESERVED   |         Payload Length        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Fragment Number        |        Total Fragments        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Initialization Vector                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                     Encrypted content                         ~
+               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               |             Padding (0-255 octets)            |
+-+-+-+-+-+-+-+-+                               +-+-+-+-+-+-+-+-+
|                                               |  Pad Length   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                    Integrity Checksum Data                    ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
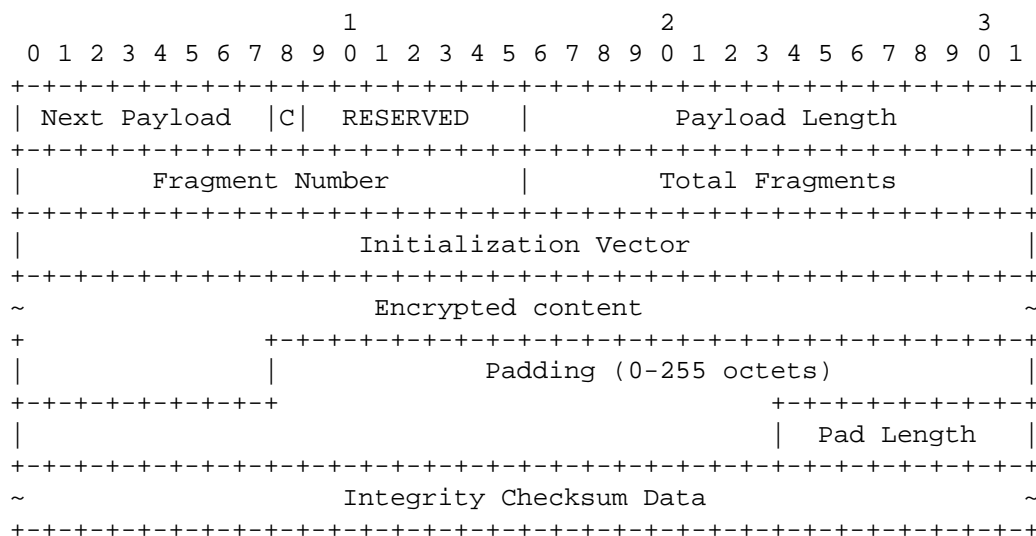
                     Encrypted Fragment Payload

   o Next Payload (1 octet) - in the very first fragment (with Fragment
      Number equal to 1), this field MUST be set to the payload type of
      the first inner payload (the same as for the Encrypted payload).
      In the rest of the Fragment messages (with Fragment Number greater
      than 1), this field MUST be set to zero.

   o Fragment Number (2 octets, unsigned integer) - current Fragment
      message number, starting from 1. This field MUST be less than or
      equal to the next field (Total Fragments). This field MUST NOT be
      zero.

   o Total Fragments (2 octets, unsigned integer) - number of Fragment
      messages into which the original message was divided. This field
      MUST NOT be zero. With PMTU discovery, this field plays an
      additional role. See Section 2.5.2 for details.

The other fields are identical to those specified in Section 3.14 of
[RFC7296].

When prepending the IKE header to the IKE Fragment messages, it MUST
be taken intact from the original message, except for the Length and
Next Payload fields. The Length field is adjusted to reflect the
length of the IKE Fragment message being constructed, and the Next
Payload field is set to the payload type of the first payload in that
message (in most cases, it will be the Encrypted Fragment payload).
After prepending the IKE header and all payloads that possibly
precede the Encrypted payload in the original message (if any; see
Section 2.5.3), the resulting messages are sent to the peer.

Below is an example of fragmenting a message.

```
HDR(MID=n), SK(NextPld=PLD1) {PLD1 ... PLDN}
```

                         Original Message


```
HDR(MID=n), SKF(NextPld=PLD1, Frag#=1, TotalFrags=m) {...},
HDR(MID=n), SKF(NextPld=0, Frag#=2, TotalFrags=m) {...},
...
HDR(MID=n), SKF(NextPld=0, Frag#=m, TotalFrags=m) {...}
```

                       IKE Fragment Messages

2.5.1. Selecting Fragment Size

When splitting the content of an Encrypted payload into chunks, the
sender SHOULD choose their size so that the resulting IP datagrams
will be smaller than some fragmentation threshold. Implementations
may calculate the fragmentation threshold using various sources of
information.

If the sender has information about the PMTU size, it SHOULD use it.
The responder in the exchange may use the maximum size of the
received IKE Fragment message IP datagrams as a threshold when
constructing a fragmented response. Successful completion of
previous exchanges (including those exchanges that cannot employ IKE
fragmentation, e.g., IKE_SA_INIT) may be an indication that the
fragmentation threshold can be set to the size of the largest message
of those messages already sent.

Otherwise, for messages to be sent over IPv6, it is RECOMMENDED that
a value of 1280 bytes as a maximum IP datagram size be used
([RFC2460]). For messages to be sent over IPv4, it is RECOMMENDED
that a value of 576 bytes as a maximum IP datagram size be used. The

presence of tunnels on the path may reduce these values.
Implementations may use other values if they are appropriate in the
current environment.

According to [RFC0791], the minimum IPv4 datagram size that is
guaranteed not to be further fragmented is 68 bytes, but it is
generally impossible to use such a small value for the solution
described in this document. Using 576 bytes is a compromise -- the
value is large enough for the presented solution and small enough to
avoid IP fragmentation in most situations. Several other UDP-based
protocols (Syslog, DNS, etc.) use 576 bytes as a safe low limit for
IP datagram size.

See Appendix B for correlation between IP datagram size and Encrypted
payload content size.

2.5.2. PMTU Discovery

The amount of traffic that the IKE endpoint produces during the
lifetime of an IKE SA is fairly modest -- it is usually below 100 KB
within a period of several hours. Most of this traffic consists of
relatively short messages -- usually below several hundred bytes. In
most cases, the only time when IKE endpoints exchange messages of
several KB in size is IKE SA establishment, and often each endpoint
sends exactly one such message.

For the reasons articulated above, implementing PMTU discovery in IKE
is OPTIONAL. It is believed that using the values recommended in
Section 2.5.1 as a fragmentation threshold will be sufficient in most
cases. Using these values could lead to suboptimal fragmentation,
but it is acceptable given the amount of traffic IKE produces.
Implementations may support PMTU discovery if there are good reasons
to do it (for example, if they are intended to be used in
environments where the MTU size might be less than the values listed
in Section 2.5.1).

PMTU discovery in IKE follows recommendations given in Section 10.4
of [RFC4821] with some modifications, induced by the distinctive
features of IKE listed above. The difference is that the PMTU search
is performed downward, while in [RFC4821] it is performed upward.
The reason for this change is that IKE usually sends large messages
only when the IKE SA is being established, and in many cases there is
only one such message. If the probing were performed upward, this
message would be fragmented using the smallest allowable threshold,
and usually all other messages are small enough to avoid IP
fragmentation, so continued probing would be of little value.

It is the initiator of the exchange who performs PMTU discovery.
This is done by probing several values of fragmentation threshold.
Implementations MUST be prepared to probe in every exchange that
utilizes IKE fragmentation to deal with possible changes in path MTU
over time. While doing probes, it MUST start from larger values and
refragment the original message, using the next smaller value of the
threshold if it did not receive a response in a reasonable time after
several retransmissions. The exact number of retransmissions and
length of timeouts are not covered in this specification because they
do not affect interoperability. However, the timeout interval is
supposed to be relatively short, so that unsuccessful probes would
not delay IKE operations too much. Performing a few retries within
several seconds for each probe seems appropriate, but different
environments may require different rules. When starting a new probe,
the node MUST reset its retransmission timers so that if it employs
exponential back-off the timers will start over. After reaching the
smallest allowed value for the fragmentation threshold, an
implementation MUST continue retransmitting until the exchange either
completes or times out using some timeout interval as discussed in
Section 2.4 of [RFC7296].

PMTU discovery in IKE is supposed to be coarse-grained, i.e., it is
expected that a node will try only a few fragmentation thresholds in
order to minimize delays caused by unsuccessful probes. If path MTU
information is not yet available, the endpoint may use the link MTU
size when it starts probing. In subsequent exchanges, the node
should start with the current value of the fragmentation threshold.

If an implementation is capable of receiving ICMP error messages, it
can additionally utilize classic PMTU discovery methods, as described
in [RFC1191] and [RFC1981]. In particular, if the initiator receives
a Packet Too Big error in response to the probe, and it contains a
smaller value than the current fragmentation threshold, then the
initiator SHOULD stop retransmitting the probe and SHOULD select a
new value for the fragmentation threshold that is less than or equal
to the value from the ICMP message and meets the requirements listed
below.

In the case of PMTU discovery, the Total Fragments field is used to
distinguish between different sets of fragments, i.e., the sets that
were created by fragmenting the original message using different
fragmentation thresholds. Since the sender starts from larger
fragments and then makes them smaller, the value in the Total
Fragments field increases with each new probe. When selecting the
next smaller value for the fragmentation threshold, the sender MUST
ensure that the value in the Total Fragments field is really
increased. This requirement should not be a problem for the sender,
because PMTU discovery in IKE is supposed to be coarse-grained, so

the difference between previous and next fragmentation thresholds
should be significant anyway. The need to distinguish between the
sets is vital for the receiver, since receiving a valid fragment from
a newer set means that it has to start the reassembly process over
and not mix fragments from different sets.

2.5.3. Fragmenting Messages Containing Unprotected Payloads

Currently, there are no IKEv2 exchanges that define messages,
containing both unprotected payloads and payloads, that are protected
by the Encrypted payload. However, IKEv2 does not prohibit such
construction. If some future IKEv2 extension defines such a message
and it needs to be fragmented, all unprotected payloads MUST be
placed in the first fragment (with the Fragment Number field equal to
1), along with the Encrypted Fragment payload, which MUST be present
in every IKE Fragment message and be the last payload in it.

Below is an example of a fragmenting message that contains both
protected and unprotected payloads.

    HDR(MID=n), PLD0, SK(NextPld=PLD1) {PLD1 ... PLDN}

                          Original Message

    HDR(MID=n), PLD0, SKF(NextPld=PLD1, Frag#=1, TotalFrags=m) {...},
    HDR(MID=n), SKF(NextPld=0, Frag#=2, TotalFrags=m) {...},
    ...
    HDR(MID=n), SKF(NextPld=0, Frag#=m, TotalFrags=m) {...}

                       IKE Fragment Messages

Note that the size of each IP datagram bearing IKE Fragment messages
should not exceed the fragmentation threshold, including the first
one, that contains unprotected payloads. This will reduce the size
of the Encrypted Fragment payload content in the first IKE Fragment
message to accommodate all unprotected payloads. In an extreme case,
the Encrypted Fragment payload will contain no data, but it still
must be present in the message, because only its presence allows the
receiver to determine that the sender has used IKE fragmentation.

2.6. Receiving IKE Fragment Message

The receiver identifies the IKE Fragment message by the presence of
an Encrypted Fragment payload in it. In most cases, it will be the
first and only payload in the message; however, this may not be true
for some hypothetical IKE exchanges (see Section 2.5.3).

Upon receiving the IKE Fragment message, the following actions are
performed:

o Check message validity - in particular, check whether the values
   in the Fragment Number and the Total Fragments fields in the
   Encrypted Fragment payload are valid. The following tests need to
   be performed.

   * check that the Fragment Number and the Total Fragments fields
      contain non-zero values

   * check that the value in the Fragment Number field is less than
      or equal to the value in the Total Fragments field

   * if reassembling has already started, check that the value in
      the Total Fragments field is equal to or greater than the Total
      Fragments field in the fragments that have already been stored
      in the reassembling queue

   If any of these tests fail, the message MUST be silently
   discarded.

o Check that this IKE Fragment message is new for the receiver and
   not a replay. If an IKE Fragment message with the same Message
   ID, Fragment Number, and Total Fragments fields is already present
   in the reassembling queue, this message is considered a replay and
   MUST be silently discarded.

o Verify IKE Fragment message authenticity by checking the Integrity
   Check Value (ICV) in the Encrypted Fragment payload. If the ICV
   check fails, the message MUST be silently discarded.

o If reassembling is not finished yet and the Total Fragments field
   in the received fragment is greater than the Total Fragments field
   in those fragments that are in the reassembling queue, the
   receiver MUST discard all received fragments and start the
   reassembly process over with just the received IKE Fragment
   message.

o Store the message in the reassembling queue waiting for the rest
   of the fragments to arrive.

When all IKE Fragment messages (as indicated in the Total Fragments
field) are received, the decrypted content of all Encrypted Fragment
payloads is merged together to form the content of the original
Encrypted payload and, therefore, along with the IKE header and

unprotected payloads (if any), the original message. Then, it is
processed as if it was received, verified, and decrypted as a regular
IKE message.

If the receiver does not get all IKE fragments needed to reassemble
the original message within a timeout interval, it MUST discard all
IKE Fragment messages received so far for the exchange. The next
actions depend on the role of the receiver in the exchange.

o The initiator acts as described in Section 2.1 of [RFC7296]. It
   either retransmits the fragmented request message or deems the IKE
   SA to have failed and deletes it. The number of retransmits and
   length of timeouts for the initiator are not covered in this
   specification, since they are assumed to be the same as in a
   regular IKEv2 exchange and are discussed in Section 2.4 of
   [RFC7296].

o The responder in this case acts as if no request message was
   received. It would delete any memory of the incomplete request
   message and not treat it as an IKE SA failure. It is RECOMMENDED
   that the reassembling timeout for the responder be equal to the
   time interval that the implementation waits before completely
   giving up when acting as the initiator of an exchange.
   Section 2.4 of [RFC7296] gives recommendations for selecting this
   interval. Implementations can use a shorter timeout to conserve
   memory.

2.6.1. Replay Detection and Retransmissions

According to Section 2.2 of [RFC7296], the Message ID is used, in
particular, to identify retransmissions of IKE messages. Each
request or response message, sent by either side, must have a unique
Message ID, or be considered a retransmission otherwise. This logic
has already been updated by [RFC6311], which deliberately allows any
number of messages with zero Message ID. This document also updates
this logic for those situations where IKE fragmentation is in use.

If an incoming message contains an Encrypted Fragment payload, the
values of the Fragment Number and Total Fragments fields MUST be used
along with the Message ID to detect retransmissions and replays.

If the responder receives a retransmitted fragment of a request when
it has already processed that request and has sent back a response,
that event MUST only trigger a retransmission of the response message
(fragmented or not) if the Fragment Number field in the received
fragment is set to 1; otherwise, it MUST be ignored.

3. Interaction with Other IKE Extensions

   IKE fragmentation is compatible with most IKE extensions, such as IKE
   Session Resumption ([RFC5723]), the Quick Crash Detection Method
   ([RFC6290]), and so on. It neither affects their operation nor is
   affected by them. It is believed that IKE fragmentation will also be
   compatible with future IKE extensions, if they follow general
   principles of formatting, sending, and receiving IKE messages, as
   described in [RFC7296].

   When IKE fragmentation is used with IKE Session Resumption
   ([RFC5723]), messages of an IKE_SESSION_RESUME exchange cannot be
   fragmented, since they do not contain an Encrypted payload. These
   messages may be large due to the ticket size. To avoid IP
   fragmentation in this situation, it is recommended that smaller
   tickets be used, e.g., by utilizing a "ticket by reference" approach
   instead of "ticket by value".

   Protocol Support for High Availability of IKEv2/IPsec, described in
   [RFC6311], requires special care when deciding whether to fragment an
   IKE message or not. Since it deliberately allows any number of
   synchronization exchanges to have the same Message ID, namely zero,
   standard IKEv2 replay detection logic, based on checking the Message
   ID, is not applicable for such messages, and the receiver has to
   check message content to detect replays. When implementing IKE
   fragmentation along with [RFC6311], IKE Message ID Synchronization
   messages MUST NOT be sent fragmented, to simplify the receiver's task
   of detecting replays. Fortunately, these messages are small, and
   there is no point in fragmenting them anyway.

4. Transport Considerations

   With IKE fragmentation, if any single IKE Fragment message gets lost,
   the receiver becomes unable to reassemble the original message. So,
   in general, using IKE fragmentation implies a higher probability that
   the message will not be delivered to the peer. Although in most
   network environments the difference will be insignificant, on some
   lossy networks it may become noticeable. When using IKE
   fragmentation, implementations MAY use longer timeouts and do more
   retransmits than usual before considering the peer dead.

   Note that Fragment messages are not individually acknowledged. The
   response Fragment messages are all sent back together only when all
   fragments of the request are received, and the original request
   message is reassembled and successfully processed.

5. Security Considerations

   Most of the security considerations for IKE fragmentation are the
   same as those for the base IKEv2 protocol described in [RFC7296].
   This extension introduces the Encrypted Fragment payload to protect
   the content of an IKE Message Fragment. This allows the receiver to
   individually check the authenticity of fragments, thus protecting
   peers from a DoS attack.

   The Security Considerations section of [RFC7296] mentions a possible
   attack on IKE where an attacker could prevent an exchange from
   completing by exhausting the IP reassembly buffers. The mechanism
   described in this document allows IKE to avoid IP fragmentation and
   therefore increases its robustness to DoS attacks.

   The following attack is possible with IKE fragmentation. An attacker
   can initiate an IKE_SA_INIT exchange, complete it, compute SK_a and
   SK_e, and then send a large but still incomplete set of IKE_AUTH
   fragments. These fragments will pass the ICV check and will be
   stored in reassembly buffers, but since the set is incomplete, the
   reassembling will never succeed and eventually will time out. If the
   set is large, this attack could potentially exhaust the receiver's
   memory resources.

   To mitigate the impact of this attack, it is RECOMMENDED that the
   receiver limit the number of fragments it stores in the reassembling
   queue so that the sum of the sizes of Encrypted Fragment payload
   contents (after decryption) for fragments that are already placed
   into the reassembling queue is less than some value that is
   reasonable for the implementation. If the peer sends so many
   fragments that the above condition is not met, the receiver can
   consider this situation to be either an attack or a broken sender
   implementation. In either case, the receiver SHOULD drop the
   connection and discard all the received fragments.

   This value can be predefined, can be a configurable option, or can be
   calculated dynamically, depending on the receiver's memory load.
   Some care should be taken when selecting this value because if it is
   too small it might prevent a legitimate peer from establishing an IKE
   SA if the size of messages it sends exceeds this value. It is NOT
   RECOMMENDED for this value to exceed 64 KB because any IKE message
   before fragmentation would likely be shorter than that.

   If IKE fragments arrive in order, it is possible, but not advised,
   for the receiver to parse the beginning of the message that is being
   reassembled and extract the already-available payloads before the
   reassembly is complete. It can be dangerous to take any action based
   on the content of these payloads, because the fragments that have not

   yet been received might contain payloads that could change the
   meaning of them (or could even make the whole message invalid), and
   this can potentially be exploited by an attacker. It is important to
   address this threat by ensuring that all the fragments are received
   prior to parsing the reassembled message, as described in
   Section 2.6.

6. IANA Considerations

   This document defines a new payload in the "IKEv2 Payload Types"
   registry:

      53   Encrypted and Authenticated Fragment   SKF

   This document also defines a new Notify Message Type in the "IKEv2
   Notify Message Types - Status Types" registry:

      16430   IKEV2_FRAGMENTATION_SUPPORTED

7. References

7.1. Normative References

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

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

   [RFC6311] Singh, R., Kalyani, G., Nir, Y., Sheffer, Y., and D.
              Zhang, "Protocol Support for High Availability of IKEv2/
              IPsec", RFC 6311, July 2011,
              <http://www.rfc-editor.org/info/rfc6311>.

7.2. Informative References

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

   [RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191,
              November 1990, <http://www.rfc-editor.org/info/rfc1191>.

   [RFC1981] McCann, J., Deering, S., and J. Mogul, "Path MTU Discovery
              for IP version 6", RFC 1981, August 1996,
              <http://www.rfc-editor.org/info/rfc1981>.

   [RFC2460]   Deering, S. and R. Hinden, "Internet Protocol, Version 6
               (IPv6) Specification", RFC 2460, December 1998,
               <http://www.rfc-editor.org/info/rfc2460>.

   [RFC4787]   Audet, F. and C. Jennings, "Network Address Translation
               (NAT) Behavioral Requirements for Unicast UDP", BCP 127,
               RFC 4787, January 2007,
               <http://www.rfc-editor.org/info/rfc4787>.

   [RFC4821]   Mathis, M. and J. Heffner, "Packetization Layer Path MTU
               Discovery", RFC 4821, March 2007,
               <http://www.rfc-editor.org/info/rfc4821>.

   [RFC5282]   Black, D. and D. McGrew, "Using Authenticated Encryption
               Algorithms with the Encrypted Payload of the Internet Key
               Exchange version 2 (IKEv2) Protocol", RFC 5282,
               August 2008, <http://www.rfc-editor.org/info/rfc5282>.

   [RFC5405]   Eggert, L. and G. Fairhurst, "Unicast UDP Usage Guidelines
               for Application Designers", BCP 145, RFC 5405,
               November 2008, <http://www.rfc-editor.org/info/rfc5405>.

   [RFC5723]   Sheffer, Y. and H. Tschofenig, "Internet Key Exchange
               Protocol Version 2 (IKEv2) Session Resumption", RFC 5723,
               January 2010, <http://www.rfc-editor.org/info/rfc5723>.

   [RFC6290]   Nir, Y., Wierbowski, D., Detienne, F., and P. Sethi, "A
               Quick Crash Detection Method for the Internet Key Exchange
               Protocol (IKE)", RFC 6290, June 2011,
               <http://www.rfc-editor.org/info/rfc6290>.

   [RFC6888]   Perreault, S., Yamagata, I., Miyakawa, S., Nakagawa, A.,
               and H. Ashida, "Common Requirements for Carrier-Grade NATs
               (CGNs)", BCP 127, RFC 6888, April 2013,
               <http://www.rfc-editor.org/info/rfc6888>.

   [FRAGDROP] Jaeggli, J., Colitti, L., Kumari, W., Vyncke, E., Kaeo,
               M., and T. Taylor, "Why Operators Filter Fragments and
               What It Implies", Work in Progress, draft-taylor-v6ops-
               fragdrop-02, December 2013.

   [BLACKHOLES]
              De Boer, M. and J. Bosma, "Discovering Path MTU black
              holes on the Internet using RIPE Atlas", July 2012,
              <http://www.nlnetlabs.nl/downloads/publications/
              pmtu-black-holes-msc-thesis.pdf>.

   [DOSUDPPROT]
              Kaufman, C., Perlman, R., and B. Sommerfeld, "DoS
              protection for UDP-based protocols", ACM Conference on
              Computer and Communications Security, October 2003.

Appendix A. Design Rationale

   The simplest approach to IKE fragmentation would have been to
   fragment a message that is fully formed and ready to be sent.
   However, if a message got fragmented after being encrypted and
   authenticated, this could make a simple DoS attack possible. The
   attacker could infrequently emit forged but valid-looking fragments
   into the network, and some of these fragments would be fetched by the
   receiver into the reassembling queue. The receiver would not be able
   to distinguish forged fragments from valid ones and would only be
   able to determine that some of the received fragments were forged
   after the whole message was reassembled and its authenticity check
   failed.

   To prevent this kind of attack and also reduce vulnerability to some
   other kinds of DoS attacks, it was decided to perform fragmentation
   before applying cryptographic protection to the message. In this
   case, each Fragment message becomes individually encrypted and
   authenticated; this allows the receiver to determine forged fragments
   and not store them in the reassembling queue.

Appendix B. Correlation between IP Datagram Size and Encrypted Payload
             Content Size

   In the case of IPv4, the content size of the Encrypted Payload is
   less than the IP datagram size by the sum of the following values:

   o IPv4 header size (typically 20 bytes, up to 60 if IP options are
      present)

   o UDP header size (8 bytes)

   o non-ESP (Encapsulating Security Payload) marker size (4 bytes if
      present)

   o IKE header size (28 bytes)

   o Encrypted payload header size (4 bytes)

   o initialization vector (IV) size (variable)

   o padding and its size (at least 1 byte)

   o ICV size (variable)

   The sum may be estimated as 61..105 bytes + IV + ICV + padding.

In the case of IPv6, the content size of the Encrypted Payload is
less than the IP datagram size by the sum of the following values:

o IPv6 header size (40 bytes)

o IPv6 extension headers (optional; size varies)

o UDP header size (8 bytes)

o non-ESP marker size (4 bytes if present)

o IKE header size (28 bytes)

o Encrypted payload header size (4 bytes)

o IV size (variable)

o padding and its size (at least 1 byte)

o ICV size (variable)

If no extension header is present, the sum may be estimated as
81..85 bytes + IV + ICV + padding. If extension headers are present,
the payload content size is further reduced by the sum of the size of
the extension headers. The length of each extension header can be
calculated as 8 * (Hdr Ext Len) bytes, except for the fragment
header, which is always 8 bytes in length.

Acknowledgements

Author's Address

   Valery Smyslov
   ELVIS-PLUS
   PO Box 81
   Moscow (Zelenograd) 124460
   Russian Federation

   Phone: +7 495 276 0211
   EMail: svan@elvis.ru