Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 7, 2014

V. Smyslov
ELVIS-PLUS

November 3, 2013

IKEv2 Fragmentation
draft-ietf-ipsecme-ikev2-fragmentation-05

Abstract

This document describes the way to avoid IP fragmentation of large IKEv2 messages. This allows IKEv2 messages to traverse network devices that don't allow IP fragments to pass through.

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 May 5, 2014.

Table of Contents

1. Introduction 3
1.1. Conventions Used in This Document 3
2. Protocol details 4
2.1. Overview 4
2.2. Limitations 4
2.3. Negotiation 4
2.4. Using IKE Fragmentation 5
2.5. Fragmenting Message 6
2.5.1. Selecting Fragment Size 8
2.5.2. PMTU Discovery 8
2.5.3. Fragmenting Messages containing unencrypted Payloads 10
2.6. Receiving IKE Fragment Message 10
2.6.1. Changes in Replay Protection Logic 12
3. Interaction with other IKE extensions 13
4. Transport Considerations 14
5. Security Considerations 15
6. IANA Considerations 16
7. Acknowledgements 17
8. References 18
8.1. Normative References 18
8.2. Informative References 18
Appendix A. Design rationale 20
Appendix B. Correlation between IP Datagram size and Encrypted Payload content size 21
Author's Address 22

1. Introduction

The Internet Key Exchange Protocol version 2 (IKEv2), specified in [RFC5996], uses UDP as a transport for its messages. Most IKEv2 messages are relatively small, usually below several hundred bytes. Noticeable exception is IKE_AUTH exchange, which requires fairly large messages, up to several kbytes, especially when certificates are transferred. When IKE message size exceeds path MTU, it gets fragmented by IP level. The problem is that some network devices, specifically some NAT boxes, don't allow IP fragments to pass through. This apparently blocks IKE communication and, therefore, prevents peers from establishing IPsec SA.

Widespread deployment of Carrier-Grade NATs (CGN) introduces new challenges. RFC6888 [RFC6888] describes requirements for CGNs. It states, that CGNs must comply with Section 11 of RFC4787 [RFC4787], which requires NAT 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 some ISPs have begun to drop IP fragments, violating that requirement.

The solution to the problem described in this document is to perform fragmentation of large messages by IKE itself, replacing them by series of smaller messages. In this case the resulting IP Datagrams will be small enough so that no fragmentation on IP level will take place.

Avoiding IP fragmentation is beneficial for IKEv2 in general. Security Considerations Section of [RFC5996] mentions exhausting of the IP reassembly buffers as one of possible attacks on the protocol. In the paper [DOSUDPPROT] several aspects of attacks on IKE using IP fragmentation are discussed, and one of defenses it proposes is to perform IKE-level fragmentation, similar to the solution, described in this document.

1.1. 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 is to split large IKE message 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. For design rationale see Appendix A.

2.2. Limitations

As IKE Fragment Messages are cryptographically protected, SK_a and SK_e must already be calculated. In general, it means that original message can be fragmented if and only if it contains Encrypted Payload.

This implies that messages of the IKE_SA_INIT Exchange cannot be fragmented. In most cases this is not a problem, since 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 MODP Groups, etc.) these messages may become fairly large and get fragmented by IP level. In this case the described solution won't help.

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

Another limitation is that the minimal size of IP Datagram bearing IKE Fragment Message is about 100 bytes depending on the algorithms employed. According to [RFC0791] the minimum IP Datagram size that is guaranteed not to be further fragmented is 68 bytes. So, even the smallest IKE Fragment Messages could be fragmented by IP level in some circumstances. But such extremely small PMTU sizes are very rare in real life.

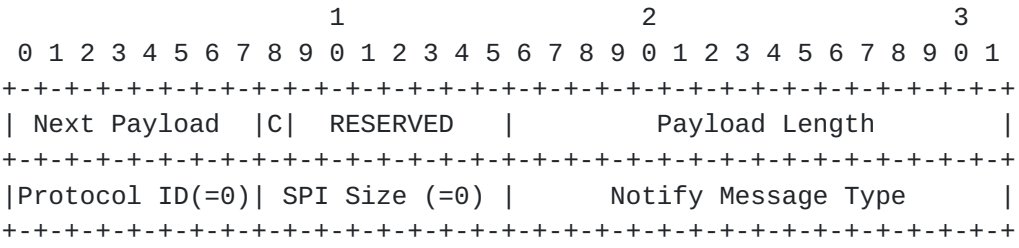
2.3. Negotiation

Initiator MAY indicate its support for IKE Fragmentation and willingness to use it by including Notification Payload of type IKE_FRAGMENTATION_SUPPORTED in IKE_SA_INIT request message. If Responder also supports this extension and is willing to use it, it includes this notification in response message.

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

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

The Notify payload is formatted as follows:



- Protocol ID (1 octet) MUST be 0.

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

- Notify Message Type (2 octets) - MUST be xxxxx, the value assigned for IKE_FRAGMENTATION_SUPPORTED by IANA.

This Notification contains no data.

2.4. Using IKE Fragmentation

IKE Fragmentation MUST NOT be used unless both peers indicated their support for it. After IKE Fragmentation is negotiated, it is up to Initiator of each Exchange, whether to use it or not. In most cases IKE Fragmentation will be used in IKE_AUTH Exchange, especially if certificates are employed. Initiator may first try to send unfragmented message and resend it fragmented only if it didn't receive response after several retransmissions, or it may always send messages fragmented (but see Section 3), or it may fragment only large messages and messages causing large responses.

In general the following guidelines are applicable for initiator:

- Initiator MAY fragment outgoing message if it has some knowledge (possibly from lower layer or from configuration) or suspicions that either request or response message will be fragmented by IP level.

- Initiator SHOULD fragment outgoing message if it has some knowledge (possibly from lower layer or from configuration) or

suspicions that either request or response message will be fragmented by IP level and IKE Fragmentation was already used in one of previous Exchanges in the context of the current IKE SA.

- Initiator SHOULD NOT fragment outgoing message if both request and response messages of the Exchange are small enough not to cause fragmentation on IP level (for example, there is no point in fragmenting Liveness Check messages).

In general the following guidelines are applicable for responder:

- Responder SHOULD send response message in the same form (fragmented or not) as corresponded request message. If it received unfragmented request message, responded with unfragmented response message and then receives fragmented retransmission of the same request, it SHOULD resend its response back to Initiator fragmented.

- Responder MAY respond to unfragmented message with fragmented response if it has some knowledge (possibly from lower layer or from configuration) or suspicions that response message will be fragmented by IP level.

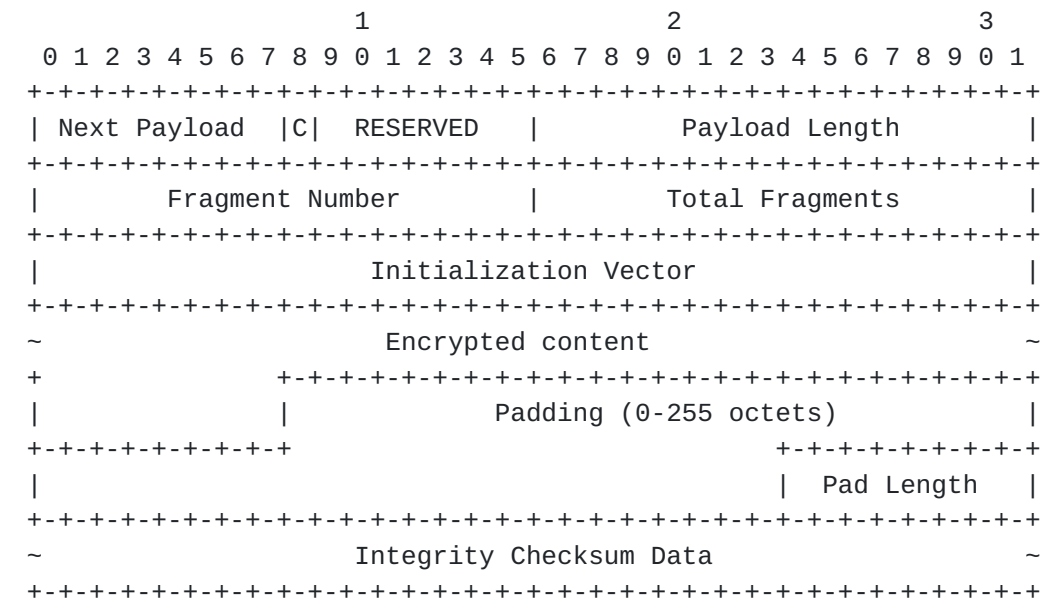
- Responder MAY respond to fragmented message with unfragmented response if the size of the response message is less than the smallest fragmentation threshold, supported by Responder (for example, there is no point in fragmenting Liveness Check messages).

2.5. Fragmenting Message

Message to be fragmented MUST contain Encrypted Payload. For the purpose of IKE Fragment Messages construction original (unencrypted) content of Encrypted Payload is split into chunks. The content is treated as a binary blob and is split regardless of inner Payloads boundaries. Each of resulting chunks is treated as an original content of Encrypted Fragment Payload and is then encrypted and authenticated. Thus, the Encrypted Fragment Payload contains a chunk of the original content of Encrypted Payload in encrypted form. The cryptographic processing of Encrypted Fragment Payload is identical to Section 3.14 of [RFC5996], as well as documents updating it for particular algorithms or modes, such as [RFC5282].

The Encrypted Fragment Payload, similarly to the Encrypted 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 XXX (TBA by IANA).



Encrypted Fragment Payload

- Next Payload (1 octet) - in the very first fragment MUST be set to Payload Type of the first inner Payload (similarly to the Encrypted Payload). In the rest fragments MUST be set to zero.

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

- Total Fragments (2 octets) - number of fragments original message was divided into. With PMTU discovery this field plays additional role. See Section 2.5.2 for details. This field MUST NOT be zero.

The other fields are identical to those specified in Section 3.14 of [RFC5996].

When prepending IKE Header, Length field MUST be adjusted to reflect the length of constructed message and Next Payload field MUST reflect payload type of the first Payload in the constructed message (that in most cases will be Encrypted Fragment Payload). All newly constructed messages MUST retain the same Message ID as original message. After prepending IKE Header and possibly any of Payloads that precedes Encrypted Payload in original message (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 content of Encrypted into chunks sender SHOULD chose size of those chunks so, that resulting IP Datagram size not exceed some fragmentation threshold - be small enough to avoid IP fragmentation.

If sender has some knowledge about PMTU size it MAY use it. If sender is a Responder in the Exchange and it has received fragmented request, it MAY use maximum size of received IKE Fragment Message IP Datagrams as threshold when constructing fragmented response.

Otherwise for messages to be sent over IPv6 it is RECOMMENDED to use value 1280 bytes as a maximum IP Datagram size ([RFC2460]). For messages to be sent over IPv4 it is RECOMMENDED to use value 576 bytes as a maximum IP Datagram size. Presence of tunnels on the path may reduce these values.

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 small value for 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 protocol assume the value 576 bytes as a safe low limit for IP datagrams size (Syslog, DNS, etc.). Sender MAY use other values if they are appropriate.

See Appendix B for correlation between IP Datagram size and Encrypted Payload content size.

2.5.2. PMTU Discovery

Initiator MAY try to discover path MTU by probing several values of fragmentation threshold. While doing probes, node MUST start from larger values and refragment message with next smaller value if it doesn't receive response in a reasonable time after several retransmissions. This time is supposed to be relatively short, so

that node could make all desired probes before exchange times out. When starting new probe (with smaller threshold) node MUST reset its retransmission timers so, that if it employs exponential back-off, the timers start over. After reaching the smallest allowed value for fragmentation threshold implementation MUST continue probing using it untill either exchange completes ot times out.

PMTU discovery in IKE is supposed to be coarse-grained, i.e. it is expected, that node will try only few fragmentation thresholds, in order to minimize possible IKE SA establishment delay. In a corner case, when host will use only one value, PMTU discovery will effectively be disabled. In most cases PMTU discovery will not be needed, as using values, recommended in section Section 2.5.1, should suffice. It is expected, that PMTU discovery may be useful in environments where PMTU size are smaller, than those listed in Section 2.5.1, for example due to the presence of intermediate tunnels.

PMTU discovery in IKE follows recommendations, given in Section 10.4 of RFC4821 [RFC4821] with some differences, induced by the specialities of IKE. In particular:

- Unlike classical PMTUD [RFC1191] and PLMTUD [RFC4821] the goal of Path MTU discovery in IKE is not to find the largest size of IP packet, that will not be fragmented en route, but to find any reasonal size, probably far from optimal.

- There is no goal to completely disallow IP fragmentation until its presence leads to inability IKE to communicate (e.g. when IP fragments are dropped)

- IKE usually sends large messages only in IKE_AUTH exchange, i.e. once per IKE SA. Most of other messages will have size below several hundred bytes. Performing full PMTUD for sending exactly one large message is inefficient.

In case of PMTU discovery Total Fragments field is used to distinguish between different sets of fragments, i.e. the sets that were obtained by fragmenting original message using different fragmentation thresholds. As sender will start from larger fragments and then make them smaller, the value in Total Fragments field will increase with each new try. When selecting next smaller value of fragmentation threshold, sender MUST ensure that the value in Total Fragments field is really increased. This requirement should not become a problem for the sender, as PMTU discovery in IKE is supposed to be coarse-grained, so difference between previous and next fragmentation thresholds will be significant anyway. The necessity to distinguish between the sets is vital for receiver as receiving

any valid fragment from newer set will mean that it have to start reassembling over and not to mix fragments from different sets.

2.5.3. Fragmenting Messages containing unencrypted Payloads

Currently no one of IKEv2 Exchanges defines messages, containing both unencrypted payloads and payloads, protected by Encrypted Payload. But IKEv2 doesn't forbid such messages. If some future IKEv2 extension defines such a message and it needs to be fragmented, all unprotected payloads MUST be in the first fragment, along with Encrypted Fragment Payload, which MUST be present in any IKE Fragment Message.

Below is an example of fragmenting message, containing both encrypted and unencrypted 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 fragmentation threshold, including the very first, which contains unprotected Payloads. This will reduce the size of Encrypted Fragment Payload content in the first IKE Fragment Message to accommodate unprotected Payloads. In extreme cases Encrypted Fragment Payload will contain no data, but it is still MUST be present in the message, because only its presence allows receiver to distinguish IKE Fragment Message from regular IKE message.

2.6. Receiving IKE Fragment Message

Receiver identifies IKE Fragment Message by the presence of Encrypted Fragment Payload in it. Note, that it is possible for this payload to be not the first (and the only) payload in the message (see Section 2.5.3). But for all currently defined IKEv2 exchanges this payload will be the first and the only payload in the message.

Upon receiving IKE Fragment Message the following actions are performed:

- Check message validity - in particular, check whether values of Fragment Number and Total Fragments in Encrypted Fragment Payload are valid. The following tests need to be performed.
  - check that Fragment Number and Total Fragments fields are non-zero
  - check that Fragment Number field is less than or equal to Total Fragments field
  - if reassembling has already started, check that Total Fragments field is equal to or greater than Total Fragments field in fragments, that have already received

  If any of this tests fails message MUST be silently discarded.

- Check, that this IKE Fragment Message is new for the receiver and not a replay. If IKE Fragment message with the same Message ID, same Fragment Number and same Total Fragments fields was already received and successfully processed, this message is considered a replay and MUST be silently discarded.

- Verify IKE Fragment Message authenticity by checking ICV in Encrypted Fragment Payload. If ICV check fails message MUST be silently discarded.

- If reassembling isn't finished yet and Total Fragments field in received IKE Fragment Message is greater than this field in previously received fragments, receiver MUST discard all received fragments and start reassembling over with just received IKE Fragment Message.

- Store message in the list waiting for the rest of fragments to arrive.

When all IKE Fragment Messages (as indicated in the Total Fragments field) are received, content of their already decrypted Encrypted Fragment Payloads is merged together to form content of original Encrypted Payload, and, therefore, along with IKE Header and unencrypted Payloads (if any), original message. Then it is processed as if it was received, verified and decrypted as regular unfragmented message.

If receiver does't get all IKE Fragment Messages needed to reassemble original Message for some Exchange within a timeout interval, it acts according with Section 2.1 of [RFC5996], i.e. retransmits the fragmented request Message (in case of Initiator) or deems Exchange to have failed. If Exchange is abandoned, all received so far IKE

Fragment Messages for that Exchange MUST be discarded.

2.6.1. Changes in Replay Protection Logic

According to [RFC5996] IKEv2 MUST reject message with the same Message ID as it has seen before (taking into consideration Response bit). 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: if message contains Encrypted Fragment Payload, the values of Fragment Number and Total Fragments fields from this payload MUST be used along with Message ID to detect retransmissions and replays.

If Responder receives IKE Fragment Message after it received, successfully verified and processed regular message with the same Message ID, it means that response message didn't reach Initiator and it activated IKE Fragmentation. If Fragment Number in Encrypted Fragment Payload in this message is equal to 1, Responder MUST fragment its response and retransmit it back to Initiator in fragmented form.

If Responder receives a replay IKE Fragment Message for already reassembled, verified and processed fragmented message, it MUST retransmit response back to Initiator, but only if Fragment Number field in Encrypted Fragment Payload is equal to 1 and MUST silently discard received message otherwise. If Total Fragments field in received IKE Fragment Message is greater than in IKE Fragment Messages that already processed fragmented message was reassembled from, Responder MAY refragment its response message using smaller fragmentation threshold before resending it back to Initiator. In this case Total Fragments field in new IKE Fragment Messages MUST be greater than in previously sent IKE Fragment Messages.

If Initiator doesn't receive any of response IKE Fragment Messages withing a timeout interval, it MAY refragment request Message using smaller fragmentation threshold before retransmitting it (see Section 2.5.1). In this case Total Fragments field in new IKE Fragment Messages MUST be greater than in previously sent IKE Fragment Messages. Alternatively, if Initiator does receive some (but not all) of response IKE Fragment Messages, it MAY retransmit only the first of request IKE Fragment Messages, where Fragment Number field is equal to 1.

3. Interaction with other IKE extensions

IKE Fragmentation is compatible with most of defined IKE extensions, like IKE Session Resumption [RFC5723], Quick Crash Detection Method [RFC6290] and so on. It neither affect their operation, nor is affected by them. It is believed that IKE Fragmentation will also be compatible with most future IKE extensions, if they follow general principles of formatting, sending and receiving IKE messages, described in [RFC5996].

When IKE Fragmentation is used with IKE Session Resumption [RFC5723], messages of IKE_SESSION_RESUME Exchange cannot be fragmented as they don't contain Encrypted Payload. These messages may be large due to ticket size. If this is the case the described solution won't help. To avoid IP Fragmentation in this situation it is recommended to use smaller tickets, e.g. by utilizing "ticket by reference" approach instead of "ticket by value".

One exception that requires a special care is [RFC6311] - Protocol Support for High Availability of IKEv2. As it deliberately allows any number of synchronization Exchanges to have the same Message ID - zero, standard replay detection logic, based on checking Message ID is not applicable for such messages, and 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 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 get lost, receiver becomes unable to reassemble original Message. So, in general, using IKE Fragmentation implies higher probability for the Message not to 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 before considering peer dead.

Note that Fragment Messages are not individually acknowledged. The response Fragment Messages are sent back all together only when all fragments of request are received, 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 base IKEv2 protocol described in [RFC5996]. This extension introduces Encrypted Fragment Payload to protect content of IKE Message Fragment. This allows receiver to individually check authenticity of fragments, thus protecting peers from DoS attack.

Security Considerations Section of [RFC5996] mentions possible attack on IKE by exhausting of the IP reassembly buffers. The mechanism, described in this document, allows IKE to avoid IP-fragmentation and therefore increases its robustness to DoS attacks.

6. IANA Considerations

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

<TBA>	Encrypted Fragment Payload	SKF

This document also defines new Notify Message Types in the "Notify Messages Types - Status Types" registry:

<TBA>	IKE_FRAGMENTATION_SUPPORTED

7. Acknowledgements

The author would like to thank Tero Kivinen, Yoav Nir, Paul Wouters, Yaron Sheffer and others for their reviews and valueable comments.

8. References

8.1. Normative References

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

[RFC5996] Kaufman, C., Hoffman, P., Nir, Y., and P. Eronen, "Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 5996, September 2010.

[RFC6311] Singh, R., Kalyani, G., Nir, Y., Sheffer, Y., and D. Zhang, "Protocol Support for High Availability of IKEv2/ IPsec", RFC 6311, July 2011.

8.2. Informative References

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

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

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

[RFC4787] Audet, F. and C. Jennings, "Network Address Translation (NAT) Behavioral Requirements for Unicast UDP", BCP 127, RFC 4787, January 2007.

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

[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.

[RFC5723] Sheffer, Y. and H. Tschofenig, "Internet Key Exchange Protocol Version 2 (IKEv2) Session Resumption", RFC 5723, January 2010.

[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.

[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.

[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 the IKE fragmentation would have been to fragment message that is fully formed and ready to be sent. But if message got fragmented after being encrypted and authenticated, this could open a possibility for a simple Denial of Service attack. The attacker could infrequently emit forged but looking valid fragments into the network, and some of these fragments would be fetched by receiver into the reassempling queue. Receiver could not distinguish forged fragments from valid ones and could only determine that some of received fragments were forged when the whole message got reassembled and check for its authenticity failed.

To prevent this kind of attack and also to reduce vulnerability to some other kinds of DoS attacks it was decided to make fragmentation before applying cryptographic protection to the message. In this case each Fragment Message becomes individually encrypted and authenticated, that allows receiver to determine forgeg fragments and not to fetch them into the reassempling queue.

Appendix B. Correlation between IP Datagram size and Encrypted Payload content size

For IPv4 Encrypted Payload content size is less than IP Datagram size by the sum of the following values:

- IPv4 header size (typically 20 bytes, up to 60 if IP options are present)
- UDP header size (8 bytes)
- non-ESP marker size (4 bytes if present)
- IKE Header size (28 bytes)
- Encrypted Payload header size (4 bytes)
- IV size (varying)
- padding and its size (at least 1 byte)
- ICV size (varying)

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

For IPv6 this estimation is difficult as there may be varying IPv6 Extension headers included.

Author's Address

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

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