Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 12, 2014

V. Smyslov
ELVIS-PLUS

June 10, 2014

IKEv2 Fragmentation
draft-ietf-ipsecme-ikev2-fragmentation-10

Abstract

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

Table of Contents

1. Introduction

1.1. Problem description

The Internet Key Exchange Protocol version 2 (IKEv2), specified in [IKEv2], 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, do not allow IP fragments to pass through. This apparently blocks IKE communication and, therefore, prevents peers from establishing IPsec SA. Section 2 of [IKEv2] discusses the impact of IP fragmentation on IKEv2 and acknowledges this problem.

Widespread deployment of Carrier-Grade NATs (CGN) introduces new challenges. [RFC6888] describes requirements for CGNs. It states, that CGNs must comply with Section 11 of [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 more and more ISPs use equipment that drop IP fragments, violating this requirement.

Security researchers have found and continue to find attack vectors that rely on IP fragmentation. For these reasons, and also 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 either by deficiency of network devices or by 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, 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.

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 it is recommended in Section 3.2 of [RFC5405]. Avoiding IP fragmentation would be beneficial for IKEv2 in general. Security Considerations Section of [IKEv2] mentions exhausting of the IP reassembly buffers as one of the possible attacks on the protocol. In the paper [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 similarly 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 is to split large IKEv2 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. See Appendix A for 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 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 MODP Groups, etc.) these messages may become fairly large and get fragmented by IP level. In this case the described solution will not 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 IPv4 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

The Initiator indicates its support for the IKE Fragmentation and willingness to use it by including Notification Payload of type IKEV2_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(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 SPI is present.
- o Notify Message Type (2 octets) - MUST be xxxxx, the value assigned for IKEV2_FRAGMENTATION_SUPPORTED notification.

This Notification contains no data.

2.4. **Using IKE Fragmentation**

The 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 always to send fragmented messages (but 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 the use of 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 it is responding to. 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 for IKE Fragmentation. For the purpose of IKE Fragment Messages construction original (unencrypted) content of the 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 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 Section 3.14 of [IKEv2], 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). 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 Payload Type of the first inner Payload (similarly to the Encrypted Payload). In the rest fragments MUST be set to zero.

- o 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.

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

The other fields are identical to those specified in Section 3.14 of [IKEv2].

When prepending IKE Header to the IKE Fragment Messages it MUST be taken intact from the original message, except for the Length and the Next Payload fields. The Length field is adjusted to reflect the length of the constructed message and the Next Payload field is set to the payload type of the first Payload in constructed message (in most cases it will be Encrypted Fragment Payload). After prepending IKE Header and all Payloads that possibly precede Encrypted Payload in 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 content of Encrypted Payload into chunks sender SHOULD choose their size so, that resulting IP Datagrams be smaller than some fragmentation threshold. Implementations may calculate fragmentation threshold using various sources of information.

If the Sender has information about PMTU size it SHOULD use it. The Responder in the exchange may use maximum size of received IKE Fragment Message IP Datagrams as threshold when constructing 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 fragmentation threshold can be set to the size of the largest of already sent messages.

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. Implementations may use other values if they are appropriate in 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 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.).

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

2.5.2. PMTU Discovery

The amount of traffic that IKE endpoint produces during lifetime of IKE SA is fairly modest - usually it is below one hundred kBytes 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 kBytes 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 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 it is intended to be used in environments where MTU size is possible to be less that values listed in Section 2.5.1).

PMTU discovery in IKE follows recommendations given in Section 10.4 of [RFC4821] with the difference, induced by the specialties 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 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 there would be little value to continue probing.

It is the Initiator of the exchange, who performs PMTU discovery. It 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 of path MTU over time. While doing probes, it MUST start from larger values and refragment original message using next smaller value of threshold if it did not receive 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 few retries within several seconds for each probe seems appropriate, but different environments may require different rules. When starting new probe 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 either exchange completes or times out using timeout interval from Section 2.4 of [IKEv2].

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 delays caused by unsuccessful probes. If no information about path MTU is known yet, endpoint may start probing from link MTU size. In the following exchanges node should start from the current value of fragmentation threshold.

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

In case of PMTU discovery Total Fragments field is used to distinguish between different sets of fragments, i.e. the sets that were created by fragmenting original message using different fragmentation thresholds. Since sender starts from larger fragments and then make them smaller, the value in Total Fragments field increases with each new probe. When selecting next smaller value for fragmentation threshold, sender MUST ensure that the value in 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 difference between previous and next fragmentation thresholds should be significant anyway. The necessity to distinguish between the sets is vital for receiver since receiving valid fragment from newer set means that it have to start reassembling over and not to 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, protected by 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 Fragment Number field equal to 1), along with Encrypted Fragment Payload, which MUST be present in every IKE Fragment Message and be the last payload in it.

Below is an example of fragmenting message, containing 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 fragmentation threshold, including the first one, that contains unprotected Payloads. This will reduce the size of Encrypted Fragment Payload content in the first IKE Fragment Message to accommodate all unprotected Payloads. In extreme case Encrypted Fragment Payload will contain no data, but it still must be present in the message, because only its presence allows receiver to determine that sender have 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 the 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:

- 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 Total Fragments field in the fragments that have already been stored in the reassembling queue

  If any of this tests fails the 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 is already present in the reassembling queue, 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 is not finished yet and Total Fragments field in received fragment is greater than this field in those fragments, that are in the reassembling queue, receiver MUST discard all received fragments and start reassembling over with just received IKE Fragment Message.

- Store message in the reassembling queue waiting for the rest of 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 content of original Encrypted Payload, and, therefore, along with IKE Header and unprotected Payloads (if any), original message. Then it is processed as if it was received, verified and decrypted as regular IKE message.

If 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 receiver in the exchange.

- The Initiator acts as described in Section 2.1 of [IKEv2]. It either retransmits the fragmented request Message or deems 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 regular IKEv2 exchange and are discussed in Section 2.4 of [IKEv2].

- 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. The reassembling timeout for the Responder is RECOMMENDED to be equal to the time interval that the implementation waits before completely giving up when acting as Initiator of exchange. Section 2.4 of [IKEv2] gives recommendations for selecting this interval. Implementations can use a shorter timeout to conserve memory.

2.6.1. Replay Detection and Retransmissions

According to [IKEv2] implementations 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 for the situations, when IKE Fragmentation is in use.

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

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

3. Interaction with other IKE extensions

IKE Fragmentation is compatible with most of IKE extensions, such as 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 future IKE extensions, if they follow general principles of formatting, sending and receiving IKE messages, described in [IKEv2].

When IKE Fragmentation is used with IKE Session Resumption ([RFC5723]), messages of IKE_SESSION_RESUME Exchange cannot be fragmented since they do not contain Encrypted Payload. These messages may be large due to the ticket size. 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 Protocol Support for High Availability of IKEv2/IPsec ([RFC6311]). 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 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 than usual 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 the base IKEv2 protocol described in [IKEv2]. 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 [IKEv2] 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.

The following attack is possible with IKE Fragmentation. An attacker can initiate 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 receiver limits the number of fragments it stores in 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 attack or as 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 legitimate peer to establish IKE SA if the size of messages it sends exceeds this value. It is NOT RECOMMENDED for this value to exceed 64 Kbytes 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 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 not yet received fragments 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 all the fragments are received prior to parse reassembled message, as it described in Section 2.6.

6. IANA Considerations

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

<TBA> Encrypted and Authenticated Fragment SKF

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

<TBA> IKEV2_FRAGMENTATION_SUPPORTED

7. Acknowledgements

The author would like to thank Tero Kivinen, Yoav Nir, Paul Wouters, Yaron Sheffer, Joe Touch, Derek Atkins, Ole Troan and others for their reviews and valuable comments. Thanks to Ron Bonica for contributing text to the Introduction Section. Thanks to Paul Hoffman and Barry Leiba for improving text clarity.

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.

[IKEv2] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", draft-kivinen-ipsecme-ikev2-rfc5996bis-03 (work in progress), April 2014.

[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.

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

[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.

[RFC5405] Eggert, L. and G. Fairhurst, "Unicast UDP Usage Guidelines for Application Designers", BCP 145, RFC 5405, November 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.

[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-02 (work in progress), 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 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 valid looking fragments into the network, and some of these fragments would be fetched by receiver into the reassembling 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 forged fragments and not to store them in the reassembling 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:

- o IPv4 header size (typically 20 bytes, up to 60 if IP options are present)
- 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 (varying)
- o padding and its size (at least 1 byte)
- o ICV size (varying)

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

For IPv6 Encrypted Payload content size is less than 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 (varying)
- o padding and its size (at least 1 byte)
- o ICV size (varying)

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.

Author's Address

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

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