IPsecME Working Group
Internet-Draft
Intended status: Standards Track
Expires: October 3, 2011

Y. Nir, Ed.
Check Point
D. Wierbowski
IBM
F. Detienne
P. Sethi
Cisco

April 1, 2011

A Quick Crash Detection Method for IKE
draft-ietf-ipsecme-failure-detection-08

Abstract

This document describes an extension to the IKEv2 protocol that allows for faster detection of Security Association (SA) desynchronization using a saved token.

When an IPsec tunnel between two IKEv2 peers is disconnected due to a restart of one peer, it can take as much as several minutes for the other peer to discover that the reboot has occurred, thus delaying recovery. In this text we propose an extension to the protocol, that allows for recovery immediately following the restart.

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 October 3, 2011.

Copyright Notice

Table of Contents

1. Introduction

IKEv2, as described in [RFC5996] and its predecessor RFC 4306, has a method for recovering from a reboot of one peer. As long as traffic flows in both directions, the rebooted peer should re-establish the tunnels immediately. However, in many cases the rebooted peer is a VPN gateway that protects only servers, so all traffic is inbound. In other cases, the non-rebooted peer has a dynamic IP address, so the rebooted peer cannot initiate IKE because its current IP address is unknown. In such cases, the rebooted peer will not be able to re-establish the tunnels. Section 2 describes how recovery works under RFC 5996, and explains why it may take several minutes.

The method proposed here, is to send an octet string, called a "QCD token" in the IKE_AUTH exchange that establishes the tunnel. That token can be stored on the peer as part of the IKE SA. After a reboot, the rebooted implementation can re-generate the token, and send it to the peer, so as to delete the IKE SA. Deleting the IKE SA results in a quick establishment of new IPsec tunnels. This is described in Section 3.

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].

The term "token" refers to an octet string that an implementation can generate using only the properties of a protected IKE message (such as IKE SPIs) as input. A conforming implementation MUST be able to generate the same token from the same input even after rebooting.

The term "token maker" refers to an implementation that generates a token and sends it to the peer as specified in this document.

The term "token taker" refers to an implementation that stores such a token or a digest thereof, in order to verify that a new token it receives is identical to the old token it has stored.

The term "non-volatile storage" in this document refers to a data storage module, that persists across restarts of the token maker. Examples of such a storage module include an internal disk, an internal flash memory module, an external disk and an external database. A small non-volatile storage module is required for a token maker, but a larger one can be used to enhance performance, as described in Section 8.2.

2. RFC 5996 Crash Recovery

When one peer loses state or reboots, the other peer does not get any notification, so unidirectional IPsec traffic can still flow. The rebooted peer will not be able to decrypt it, however, and the only remedy is to send an unprotected INVALID_SPI notification as described in section 3.10.1 of [RFC5996]. That section also describes the processing of such a notification:

> "If this Informational Message is sent outside the context of an IKE_SA, it should be used by the recipient only as a "hint" that something might be wrong (because it could easily be forged)."

Since the INVALID_SPI can only be used as a hint, the non-rebooted peer has to determine whether the IPsec SA, and indeed the parent IKE SA are still valid. The method of doing this is described in section 2.4 of [RFC5996]. This method, called "liveness check" involves sending a protected empty INFORMATIONAL message, and awaiting a response. This procedure is sometimes referred to as "Dead Peer Detection" or DPD.

Section 2.4 does not mandate how many times the liveness check message should be retransmitted, or for how long, but does recommend the following:

> "It is suggested that messages be retransmitted at least a dozen times over a period of at least several minutes before giving up on an SA..."

Those "at least several minutes" are a time during part of which both peers are active, but IPsec cannot be used.

Especially in the case of a reboot (rather than fail-over or administrative clearing of state), the peer does not recover immediately. Reboot, depending on the system, may take from a few seconds to a few minutes. This means that at first the peer just goes silent, i.e., does not send or respond to any messages. IKEv2 implementations can detect this situation and follow the rules given in section 2.4:

> If there has only been outgoing traffic on all of the SAs associated with an IKE SA, it is essential to confirm liveness of the other endpoint to avoid black holes. If no cryptographically protected messages have been received on an IKE SA or any of its Child SAs recently, the system needs to perform a liveness check in order to prevent sending messages to a dead peer.

[RFC5996] does not mandate any time limits, but it is possible that the peer will start liveness checks even before the other end is sending INVALID_SPI notification, as it detected that the other end is not sending any packets anymore while it is still rebooting or recovering from the situation.

This means that the several minutes recovery period is overlaping the actual recover time of the other peer, i.e., if the security gateway requires several minutes to boot up from the crash then the other peers have already finished their liveness checks before the crashing peer even has a chance to send INVALID_SPI notifications.

There are cases where the peer loses state and is able to recover immediately; in those cases it might take several minutes to recreate the IPsec SAs.

Note that the IKEv2 specification specifically gives no guidance for the number of retries or the length of timeouts, as these do not affect interoperability. This means that implementations are allowed to use the hints provided by the INVALID_SPI messages to shorten those timeouts (i.e., different environment and situation requiring different rules).

Some existing IKEv2 implementations already do that (i.e., both shorten timeouts or limit number of retries) based on these kind of hints and also start liveness checks quickly after the other end goes silent. However, see Appendix A.4 for a discussion of why this may not be enough.

3. Protocol Outline

Supporting implementations will send a notification, called a "QCD token", as described in Section 4.1 in the first IKE_AUTH exchange messages. These are the first IKE_AUTH request and final IKE_AUTH response that contain the AUTH payloads. The generation of these tokens is a local matter for implementations, but considerations are described in Section 5. Implementations that send such a token will be called "token makers".

A supporting implementation receiving such a token MUST store it (or a digest thereof) along with the IKE SA. Implementations that support this part of the protocol will be called "token takers". Section 8.1 has considerations for which implementations need to be token takers, and which should be token makers. Implementations that are not token takers will silently ignore QCD tokens.

When a token maker receives a protected IKE request message with

unknown IKE SPIs, it SHOULD generate a new token that is identical to the previous token, and send it to the requesting peer in an unprotected IKE message as described in Section 4.5.

When a token taker receives the QCD token in an unprotected notification, it MUST verify that the TOKEN_SECRET_DATA matches the token stored with the matching IKE SA. If the verification fails, or if the IKE SPIs in the message do not match any existing IKE SA, it SHOULD log the event. If it succeeds, it MUST silently delete the IKE SA associated with the IKE_SPI fields, and all dependent child SAs. This event MAY also be logged. The token taker MUST accept such tokens from any IP address and port combination, so as to allow different kinds of high-availability configurations of the token maker.

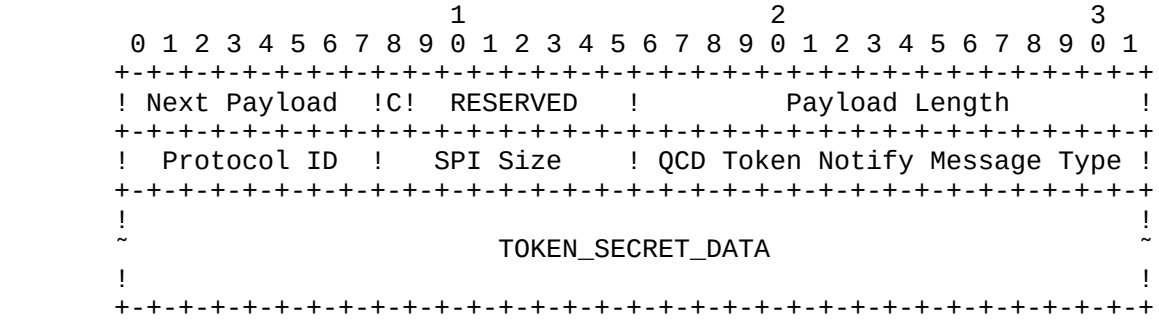
A supporting token taker MAY immediately create new SAs using an Initial exchange, or it may wait for subsequent traffic to trigger the creation of new SAs.

See Section 7 for a short discussion about this extensions's interaction with IKEv2 Session Resumption ([RFC5723]).

4. Formats and Exchanges

4.1. Notification Format

The notification payload called "QCD token" is formatted as follows:



- o Protocol ID (1 octet) MUST be 1, as this message is related to an IKE SA.
- o SPI Size (1 octet) MUST be zero, in conformance with section 3.10 of [RFC5996].
- o QCD Token Notify Message Type (2 octets) - MUST be xxxxx, the value assigned for QCD token notifications. TBA by IANA.

o TOKEN_SECRET_DATA (variable) contains a generated token as described in Section 5.

4.2. Passing a Token in the AUTH Exchange

For brevity, only the EAP version of an AUTH exchange will be presented here. The non-EAP version is very similar. The figures below are based on appendix C.3 of [RFC5996].

```
 first request       --> IDi,
                         [N(INITIAL_CONTACT)],
                         [[N(HTTP_CERT_LOOKUP_SUPPORTED)], CERTREQ+],
                         [IDr],
                         [N(QCD_TOKEN)]
                         [CP(CFG_REQUEST)],
                         [N(IPCOMP_SUPPORTED)+],
                         [N(USE_TRANSPORT_MODE)],
                         [N(ESP_TFC_PADDING_NOT_SUPPORTED)],
                         [N(NON_FIRST_FRAGMENTS_ALSO)],
                         SA, TSi, TSr,
                         [V+]

 first response      <-- IDr, [CERT+], AUTH,
                         EAP,
                         [V+]

                   / --> EAP
 repeat 1..N times |
                   \ <-- EAP

 last request        --> AUTH

 last response       <-- AUTH,
                         [N(QCD_TOKEN)]
                         [CP(CFG_REPLY)],
                         [N(IPCOMP_SUPPORTED)],
                         [N(USE_TRANSPORT_MODE)],
                         [N(ESP_TFC_PADDING_NOT_SUPPORTED)],
                         [N(NON_FIRST_FRAGMENTS_ALSO)],
                         SA, TSi, TSr,
                         [N(ADDITIONAL_TS_POSSIBLE)],
                         [V+]
```

Note that the QCD_TOKEN notification is marked as optional because it is not required by this specification that every implementation be both token maker and token taker. If only one peer sends the QCD token, then a reboot of the other peer will not be recoverable by this method. This may be acceptable if traffic typically originates

from the other peer.

In any case, the lack of a QCD_TOKEN notification MUST NOT be taken as an indication that the peer does not support this standard. Conversely, if a peer does not understand this notification, it will simply ignore it. Therefore a peer may send this notification freely, even if it does not know whether the other side supports it.

The QCD_TOKEN notification is related to the IKE SA and should follow the AUTH payload and precede the Configuration payload and all payloads related to the child SA.

4.3. Replacing Tokens After Rekey or Resumption

After rekeying an IKE SA, the IKE SPIs are replaced, so the new SA also needs to have a token. If only the responder in the rekey exchange is the token maker, this can be done within the CREATE_CHILD_SA exchange. If the initiator is a token maker, then we need an extra informational exchange.

The following figure shows the CREATE_CHILD_SA exchange for rekeying the IKE SA. Only the responder sends a QCD token.

```
request             --> SA, Ni, [KEi]

response            <-- SA, Nr, [KEr], N(QCD_TOKEN)
```

If the initiator is also a token maker, it SHOULD initiate an INFORMATIONAL exchange immediately after the CREATE_CHILD_SA exchange as follows:

```
request             --> N(QCD_TOKEN)

response            <--
```

For session resumption, as specified in [RFC5723], the situation is similar. The responder, which is necessarily the peer that has crashed, SHOULD send a new ticket within the protected payload of the IKE_SESSION_RESUME exchange. If the Initiator is also a token maker, it needs to send a QCD_TOKEN in a separate INFORMATIONAL exchange.

The INFORMATIONAL exchange described in this section can also be used if QCD tokens need to be replaced due to a key rollover. However, since token takers are required to verify at least 4 QCD tokens, this is only necessary if secret QCD keys are rolled over more than four times as often as IKE SAs are rekeyed. See Section 5.1 for an example method that uses secret keys which may require rollover.

4.4. Replacing the Token for an Existing SA

With some token generation methods, such as that described in Section 5.2, a QCD token may sometimes become invalid, although the IKE SA is still perfectly valid.

In such a case, the token maker MUST send the new token in a protected message under that IKE SA. That exchange could be a simple INFORMATIONAL, such as in the last figure in the previous section, or else it can be part of a MOBIKE INFORMATIONAL exchange such as in the following figure taken from section 2.2 of [RFC4555] and modified by adding a QCD_TOKEN notification:

```
  (IP_I2:4500 -> IP_R1:4500)
  HDR, SK { N(UPDATE_SA_ADDRESSES),
            N(NAT_DETECTION_SOURCE_IP),
            N(NAT_DETECTION_DESTINATION_IP) }  -->

                        <-- (IP_R1:4500 -> IP_I2:4500)
                            HDR, SK { N(NAT_DETECTION_SOURCE_IP),
                                 N(NAT_DETECTION_DESTINATION_IP) }

                        <-- (IP_R1:4500 -> IP_I2:4500)
                            HDR, SK { N(COOKIE2), [N(QCD_TOKEN)] }

  (IP_I2:4500 -> IP_R1:4500)
  HDR, SK { N(COOKIE2), [N(QCD_TOKEN)] }  -->
```

A token taker MUST accept such gratuitous QCD_TOKEN notifications as long as they are carried in protected exchanges. A token maker SHOULD NOT generate them unless it is no longer able to generate the old QCD_TOKEN.

4.5. Presenting the Token in an Unprotected Message

This QCD_TOKEN notification is unprotected, and is sent as a response to a protected IKE request, which uses an IKE SA that is unknown.

```
          message             --> N(INVALID_IKE_SPI), N(QCD_TOKEN)+
```

If child SPIs are persistently mapped to IKE SPIs as described in Section 8.2, a token taker may get the following unprotected message in response to an ESP or AH packet.

```
          message             --> N(INVALID_SPI), N(QCD_TOKEN)+
```

The QCD_TOKEN and INVALID_IKE_SPI notifications are sent together to support both implementations that conform to this specification and

implementations that don't. Similar to the description in section 2.21 of [RFC5996], the IKE SPI and message ID fields in the packet headers are taken from the protected IKE request.

To support a periodic rollover of the secret used for token generation, the token taker MUST support at least four QCD_TOKEN notifications in a single packet. The token is considered verified if any of the QCD_TOKEN notifications matches. The token maker MAY generate up to four QCD_TOKEN notifications, based on several generations of keys.

If the QCD_TOKEN verifies OK, the receiver MUST silently discard the IKE SA and all associated child SAs. If the QCD_TOKEN cannot be validated, a response MUST NOT be sent, and the event may be logged. Section 5 defines token verification.

5. Token Generation and Verification

No token generation method is mandated by this document. Two methods are documented in the following sub-sections, but they only serve as examples.

The following lists the requirements for a token generation mechanism:

- Tokens MUST be at least 16 octets long, and no more than 128 octets long, to facilitate storage and transmission. Tokens SHOULD be indistinguishable from random data.
- It should not be possible for an external attacker to guess the QCD token generated by an implementation. Cryptographic mechanisms such as PRNG and hash functions are RECOMMENDED.
- The token maker MUST be able to re-generate or retrieve the token based on the IKE SPIs even after it reboots.
- The method of token generation MUST be such that a collision of QCD tokens between different pairs of IKE SPI will be highly unlikely.

For verification, the token taker makes a bitwise comparison of the token stored along with the IKE SA with the token sent in the unprotected message. Multihomed takers might flip back-and-forth between several addresses, and have their tokens replaced as described in Section 4.4. To help avoid the case where the latest stored token does not match the address used after the maker lost state, the token taker MAY store several earlier tokens associated with the IKE SA, and silently discard the SA if any of them matches.

5.1. A Stateless Method of Token Generation

The following describes a stateless method of generating a token. In this case, 'stateless' means not maintaining any per-tunnel state, although there is a small amount of non-volatile storage required.

- At installation or immediately after the first boot of the token maker, 32 random octets are generated using a secure random number generator or a PRNG.
- Those 32 bytes, called the "QCD_SECRET", are stored in non-volatile storage on the machine, and kept indefinitely.
- If key rollover is required by policy, the implementation MAY periodically generate a new QCD_SECRET and keep up to 3 previous generations. When sending an unprotected QCD_TOKEN, as many as 4 notification payloads may be sent, each from a different QCD_SECRET.
- The TOKEN_SECRET_DATA is calculated as follows:

```
TOKEN_SECRET_DATA = HASH(QCD_SECRET | SPI-I | SPI-R)
```

5.2. A Stateless Method with IP addresses

This method is similar to the one in the previous section, except that the IP address of the token taker is also added to the block being hashed. This has the disadvantage that the token needs to be replaced (as described in Section 4.4) whenever the token taker changes its address.

See Section 9.2 for a discussion of a use-case for this method. When using this method, the TOKEN_SECRET_DATA field is calculated as follows:

```
TOKEN_SECRET_DATA = HASH(QCD_SECRET | SPI-I | SPI-R | IPaddr-T)
```

The IPaddr-T field specifies the IP address of the token taker. Secret rollover considerations are similar to those in the previous section.

Note that with a multi-homed token taker, the QCD token matches just one of the token taker IP addresses. Usually this is not a problem, as packets sent to the token maker come out the same IP address. If for some reason this changes, then the token maker can replace the token as described in section 4.4. If MOBIKE is used, replacing the tokens SHOULD be piggybacked on the INFORMATIONAL exchange with the UPDATE_SA_ADDRESSES notifications.

There is a corner case where the token taker begins using a new IP address (because of multi-homing, roaming or normal network operations) and the token maker loses state before replacing the token. In that case, it will send a correct QCD token, but the token taker will still have the old token. In that case the extension will not work, and the peers will revert to RFC 5996 recovery.

5.3. Token Lifetime

The token is associated with a single IKE SA, and SHOULD be deleted by the token taker when the SA is deleted or expires. More formally, the token is associated with the pair (SPI-I, SPI-R).

6. Backup Gateways

Making crash detection and recovery quick is a worthy goal, but since rebooting a gateway takes a non-zero amount of time, many implementations choose to have a stand-by gateway ready to take over as soon as the primary gateway fails for any reason. [RFC6027] describes considerations for such clusters of gateways with synchronized state, but the rest of this section is relevant even when there is no synchronized state.

If such a configuration is available, it is RECOMMENDED that the stand-by gateway be able to generate the same token as the active gateway. if the method described in Section 5.1 is used, this means that the QCD_SECRET field is identical in both gateways. This has the effect of having the crash recovery available immediately.

Note that this refers to "high availability" configurations, where only one gateway is active at any given moment. This is different from "load sharing" configurations where more than one gateway is active at the same time. For load sharing configurations, please see Section 9.2 for security considerations.

7. Interaction with Session Resumption

Session resumption, specified in [RFC5723], allows the setting up of a new IKE SA to consume less computing resources. This is particularly useful in the case of a remote access gateway that has many tunnels. A failure of such a gateway requires all these many remote access clients to establish an IKE SA either with the rebooted gateway or with a backup. This tunnel re-establishment occurs within a short period of time, creating a burden on the remote access gateway. Session resumption addresses this problem by having the clients store an encrypted derivative of the IKE SA for quick re-

establishment.

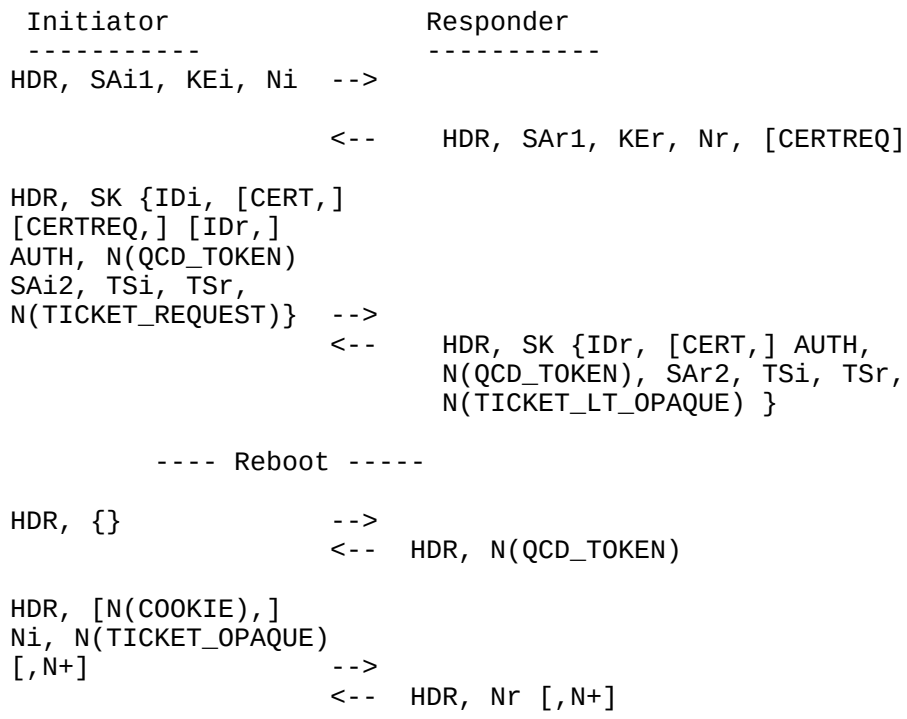
What Session Resumption does not help is the problem of detecting that the peer gateway has failed. A failed gateway may go undetected for an arbitrarily long time, because IPsec does not have packet acknowledgement, and applications cannot signal the IPsec layer that the tunnel "does not work". Section 2.4 of RFC 5996 does not specify how long an implementation needs to wait before beginning a liveness check, and only says "not recently" (see full quote in Section 2). In practice some mobile devices wait a very long time before beginning liveness check, in order to extend battery life by allowing parts of the device to remain in low-power modes.

QCD tokens provide a way to detect the failure of the peer in the case where liveness check has not yet ended (or begun).

A remote access client conforming to both specifications will store QCD tokens, as well as the Session Resumption ticket, if provided by the gateway. A remote access gateway conforming to both specifications will generate a QCD token for the client. When the gateway reboots, the client will discover this in either of two ways:

1. The client does regular liveness checks, or else the time for some other IKE exchange has come. Since the gateway is still down, the IKE exchange times out after several minutes. In this case QCD does not help.
2. Either the primary gateway or a backup gateway (see Section 6) is ready and sends a QCD token to the client. In that case the client will quickly re-establish the IPsec tunnel, either with the rebooted primary gateway or the backup gateway as described in this document.

The full combined protocol looks like this:



8. Operational Considerations

8.1. Who should implement this specification

Throughout this document, we have referred to reboot time alternatingly as the time that the implementation crashes and the time when it is ready to process IPsec packets and IKE exchanges. Depending on the hardware and software platforms and the cause of the reboot, rebooting may take anywhere from a few seconds to several minutes. If the implementation is down for a long time, the benefit of this protocol extension is reduced. For this reason critical systems should implement backup gateways as described in Section 6.

Implementing the "token maker" side of QCD makes sense for IKE implementation where protected connections originate from the peer, such as inter-domain VPNs and remote access gateways. Implementing the "token taker" side of QCD makes sense for IKE implementations where protected connections originate, such as inter-domain VPNs and remote access clients.

To clarify the this discussion:

- For remote-access clients it makes sense to implement the token taker role.
- For remote-access gateways it makes sense to implement the token maker role.
- For inter-domain VPN gateways it makes sense to implement both roles, because it can't be known in advance where the traffic originates.
- It is perfectly valid to implement both roles in any case, for example when using a single library or a single gateway to perform several roles.

In order to limit the effects of DoS attacks, a token taker SHOULD limit the rate of QCD_TOKENs verified from a particular source.

If excessive amounts of IKE requests protected with unknown IKE SPIs arrive at a token maker, the IKE module SHOULD revert to the behavior described in section 2.21 of [RFC5996] and either send an INVALID_IKE_SPI notification, or ignore it entirely.

Section 9.2 requires that token makers never send a QCD token in the clear for a valid IKE SA, and describes some configurations where this could occur. Implementations that may be installed in such configurations SHOULD automatically detect this and disable this extension in unsafe configurations, and MUST allow the user to control whether the extension is enabled or disabled.

8.2. Response to unknown child SPI

After a reboot, it is more likely that an implementation receives IPsec packets than IKE packets. In that case, the rebooted implementation will send an INVALID_SPI notification, triggering a liveness check. The token will only be sent in a response to the liveness check, thus requiring an extra round-trip.

To avoid this, an implementation that has access to enough non-volatile storage MAY store a mapping of child SPIs to owning IKE SPIs, or to generated tokens. If such a mapping is available and persistent across reboots, the rebooted implementation SHOULD respond to the IPsec packet with an INVALID_SPI notification, along with the appropriate QCD_Token notifications. A token taker SHOULD verify the QCD token that arrives with an INVALID_SPI notification the same as if it arrived with the IKE SPIs of the parent IKE SA.

However, a persistent storage module might not be updated in a timely manner, and could be populated with tokens relating to IKE SPIs that have already been rekeyed. A token taker MUST NOT take an invalid QCD Token sent along with an INVALID_SPI notification as evidence that the peer is either malfunctioning or attacking, but it SHOULD

limit the rate at which such notifications are processed.

9. Security Considerations

The extension described in this document must not reduce the security of IKEv2 or IPsec. Specifically, an eavesdropper must not learn any non-public information about the peers.

The proposed mechanism should be secure against attacks by a passive MITM (eavesdropper). Such an attacker must not be able to disrupt an existing IKE session, either by resetting the session or by introducing significant delays. This requirement is especially significant, because this document introduces a new way to reset an IKE SA.

The mechanism need not be similarly secure against an active MITM, since this type of attacker is already able to disrupt IKE sessions.

9.1. QCD Token Generation and Handling

Tokens MUST be hard to guess. This is critical, because if an attacker can guess the token associated with an IKE SA, they can tear down the IKE SA and associated tunnels at will. When the token is delivered in the IKE_AUTH exchange, it is encrypted. When it is sent again in an unprotected notification, it is not, but that is the last time this token is ever used.

An aggregation of some tokens generated by one maker together with the related IKE SPIs MUST NOT give an attacker the ability to guess other tokens. Specifically, if one taker does not properly secure the QCD tokens and an attacker gains access to them, this attacker MUST NOT be able to guess other tokens generated by the same maker. This is the reason that the QCD_SECRET in Section 5.1 needs to be sufficiently long.

The token taker MUST store the token in a secure manner. No attacker should be able to gain access to a stored token.

The QCD_SECRET MUST be protected from access by other parties. Anyone gaining access to this value will be able to delete all the IKE SAs for this token maker.

The QCD token is sent by the rebooted peer in an unprotected message. A message like that is subject to modification, deletion and replay by an attacker. However, these attacks will not compromise the security of either side. Modification is meaningless because a modified token is simply an invalid token. Deletion will only cause

the protocol not to work, resulting in a delay in tunnel re-establishment as described in Section 2. Replay is also meaningless, because the IKE SA has been deleted after the first transmission.

9.2. QCD Token Transmission

A token maker MUST NOT send a valid QCD token in an unprotected message for an existing IKE SA.

This requirement is obvious and easy in the case of a single gateway. However, some implementations use a load balancer to divide the load between several physical gateways. It MUST NOT be possible even in such a configuration to trick one gateway into sending a valid QCD token for an IKE SA which is valid on another gateway. This is true whether the attempt to trick the gateway uses the token taker's IP address or a different IP address.

IPsec Failure Detection is not applicable to deployments where the QCD secret is shared by multiple gateways and the gateways cannot assess whether the token can be legitimately sent in the clear while another gateway may actually still own the SA's. Load balancer configurations typically fall in this category. In order for a load balancing configuration of IPsec gateways to support this specification, all members MUST be able to tell whether a particular IKE SA is active anywhere in the cluster. One way to do this is to synchronize a list of active IKE SPIs among all the cluster members.

Because it includes the token taker's IP address in the token generation, the method in Section 5.2 can (under certain conditions) prevent revealing the QCD token for an existing pair of IKE SPIs to an attacker who is using a different IP address, even in a load-sharing cluster without state synchronization. That method does not prevent revealing the QCD token to an active attacker who is spoofing the token taker's IP address. Such an attacker may attempt to direct messages to a cluster member other than the member responsible for the IKE SA in an attempt to trick that gateway into sending a QCD token for a valid IKE SA. That method should not be used unless the load balancer guarantees that IKE packets from the same source IP address always go to the same cluster member.

9.3. QCD Token Enumeration

An attacker may try to attack QCD if the generation algorithm described in Section 5.1 is used. The attacker will send several fake IKE requests to the gateway under attack, receiving and recording the QCD Tokens in the responses. This will allow the attacker to create a dictionary of IKE SPIs to QCD Tokens, which can later be used to tear down any IKE SA.

Three factors mitigate this threat:

- The space of all possible IKE SPI pairs is huge: 2^128, so making such a dictionary is impractical. Even if we assume that one implementation always generates predictable IKE SPIs, the space is still at least 2^64 entries, so making the dictionary is extremely hard. To ensure this, token makers MUST generate unpredictable IKE SPIs by using a cryptographically strong pseudo-random number generator.
- Throttling the amount of QCD_TOKEN notifications sent out, as discussed in Section 8.1, especially when not soon after a crash will limit the attacker's ability to construct a dictionary.
- The methods in Section 5.1 and Section 5.2 allow for a periodic change of the QCD_SECRET. Any such change invalidates the entire dictionary.

10. IANA Considerations

IANA is requested to assign a notify message type from the status types range (16406-40959) of the "IKEv2 Notify Message Types" registry with name "QUICK_CRASH_DETECTION".

11. Acknowledgements

We would like to thank Hannes Tschofenig and Yaron Sheffer for their comments about Session Resumption.

Others who have contributed valuable comments are, in alphabetical order, Lakshminath Dondeti, Paul Hoffman, Tero Kivinen, Scott C Moonen, Magnus Nystrom, and Keith Welter.

12. Change Log

This section lists all changes in this document

NOTE TO RFC EDITOR : Please remove this section in the final RFC

12.1. Changes from draft-ietf-ipsecme-failure-detection-05

- Some clarifications suggested by Magnus Nystrom.

12.2. Changes from draft-ietf-ipsecme-failure-detection-04

- Some more rephrasing of section 9.2 based on suggestions by Tero Kivinen and Dave Wierbowski.

12.3. Changes from draft-ietf-ipsecme-failure-detection-03

- Merged section 9.4 into section 9.2.
- Multiple typos discovered by Scott Moonen, Keith Welter and Yaron.

12.4. Changes from draft-ietf-ipsecme-failure-detection-02

- Moved section 7 to Appendix A. Also changed some wording.
- Fixed some language in the "interaction with session resumption" section to say that although liveness check MUST be done, there are no time limits to how long an implementation takes before starting liveness check, or ending it.

12.5. Changes from draft-ietf-ipsecme-failure-detection-01

- Fixed the language requiring random IKE SPIs.
- Some better explanation of the reasons to choose the methods in Section 5.2 and the method in Section 5.1, to close issue #193.
- Added text to the beginning of Section 9 to accomodate issue #194.

12.6. Changes from draft-ietf-ipsecme-failure-detection-00

- Nits pointed out by Scott and Yaron.
- Pratima and Frederic are back on board.
- Changed IKEv2bis draft reference to RFC 5996.
- Resolved issues #189, #190, #191, and #192:
  * Renamed section 4.5 and removed the requirement to send an acknowledgement for the unprotected message.
  * Moved the QCD token from the last to the first IKE_AUTH request.
  * Added a MUST to Section 9.3 to require that IKE SPIs be randomly generated.
  * Changed the language in Section 8.1, to not use RFC 2119 terminology.
  * Moved the section describing why one would want the method dependant on IP addresses (in Section 5.2 from operational considerations to security considerations.

12.7. Changes from draft-nir-ike-qcd-07

- First WG version.
- Addressed Scott C Moonen’s concern about collisions of QCD tokens.
- Updated references to point to IKEv2bis instead of RFC 4306 and 4718. Also converted draft reference for resumption to RFC 5723.
- Added Dave Wiebrowski as author, and removed Pratima and Frederic.

12.8. Changes from draft-nir-ike-qcd-03 and -04

Mostly editorial changes and cleaning up.

12.9. Changes from draft-nir-ike-qcd-02

- Described QCD token enumeration, following a question by Lakshminath Dondeti.
- Added the ability to replace the QCD token for an existing IKE SA.
- Added tokens dependent on peer IP address and their interaction with MOBIKE.

12.10. Changes from draft-nir-ike-qcd-01

- Removed stateless method.
- Added discussion of rekeying and resumption.
- Added discussion of non-synchronized load-balanced clusters of gateways in the security considerations.
- Other wording fixes.

12.11. Changes from draft-nir-ike-qcd-00

- Merged proposal with draft-detienne-ikev2-recovery
- Changed the protocol so that the rebooted peer generates the token. This has the effect, that the need for persistent storage is eliminated.
- Added discussion of birth certificates.

12.12. Changes from draft-nir-qcr-00

- Changed name to reflect that this relates to IKE. Also changed from quick crash recovery to quick crash detection to avoid confusion with IFARE.
- Added more operational considerations.
- Added interaction with IFARE.
- Added discussion of backup gateways.

13. References

13.1. Normative References

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

[RFC4555] Eronen, P., "IKEv2 Mobility and Multihoming Protocol (MOBIKE)", RFC 4555, June 2006.

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

13.2. Informative References

[RFC5723] Sheffer, Y. and H. Tschofenig, "IKEv2 Session Resumption", RFC 5723, January 2010.

[RFC6027] Nir, Y., Ed., "IPsec Cluster Problem Statement", RFC 6027, October 2010.

[recovery] Detienne, F., Sethi, P., and Y. Nir, "Safe IKE Recovery", draft-detienne-ikev2-recovery (work in progress), July 2009.

Appendix A. The Path Not Taken

A.1. Initiating a new IKE SA

Instead of sending a QCD token, we could have the rebooted implementation start an Initial exchange with the peer, including the INITIAL_CONTACT notification. This would have the same effect, instructing the peer to erase the old IKE SA, as well as establishing a new IKE SA with fewer rounds.

The disadvantage here, is that in IKEv2 an authentication exchange MUST have a piggy-backed Child SA set up. Since our use case is such that the rebooted implementation does not have traffic flowing to the peer, there are no good selectors for such a Child SA.

Additionally, when authentication is asymmetric, such as when EAP is used, it is not possible for the rebooted implementation to initiate IKE.

A.2. SIR

Another proposal that was considered for this work item is the SIR extension, which is described in [recovery]. Under that proposal, the non-rebooted peer sends a non-protected query to the possibly rebooted peer, asking whether the IKE SA exists. The peer replies with either a positive or negative response, and the absence of a positive response, along with the existence of a negative response is taken as proof that the IKE SA has really been lost.

The working group preferred the QCD proposal to this one.

A.3. Birth Certificates

Birth Certificates is a method of crash detection that has never been formally defined. Bill Sommerfeld suggested this idea in a mail to the IPsec mailing list on August 7, 2000, in a thread discussing methods of crash detection:

> If we have the system sign a "birth certificate" when it reboots (including a reboot time or boot sequence number), we could include that with a "bad spi" ICMP error and in the negotiation of the IKE SA.

We believe that this method would have some problems. First, it requires Alice to store the certificate, so as to be able to compare the public keys. That requires more storage than does a QCD token. Additionally, the public-key operations needed to verify the self-signed certificates are more expensive for Alice.

We believe that a symmetric-key operation such as proposed here is more light-weight and simple than that implied by the Birth Certificate idea.

A.4. Reducing Liveness Check Length

Some implementations require fewer retransmissions over a shorter period of time for cases of liveness check started because of an INVALID_SPI or INVALID_IKE_SPI notification.

We believe that the default retransmission policy should represent a good balance between the need for a timely discovery of a dead peer, and a low probability of false detection. We expect the policy to be set to take the shortest time such that this probability achieves a certain target. Therefore, we believe that reducing the elapsed time and retransmission count may create an unacceptably high probability of false detection, and this can be triggered by a single INVALID_IKE_SPI notification.

Additionally, even if the retransmission policy is reduced to, say, one minute, it is still a very noticeable delay from a human perspective, from the time that the gateway has come up (i.e., is able to respond with an INVALID_SPI or INVALID_IKE_SPI notification) and until the tunnels are active, or from the time the backup gateway has taken over until the tunnels are active. The use of QCD tokens can reduce this delay.

Authors' Addresses

Yoav Nir (editor)
Check Point Software Technologies Ltd.
5 Hasolelim st.
Tel Aviv 67897
Israel

Email: ynir@checkpoint.com

David Wierbowski
International Business Machines
1701 North Street
Endicott, New York 13760
United States

Email: wierbows@us.ibm.com

Frederic Detienne
Cisco Systems, Inc.
De Kleetlaan, 7
Diegem B-1831
Belgium

Phone: +32 2 704 5681
Email: fd@cisco.com

Pratima Sethi
Cisco Systems, Inc.
O'Shaugnessy Road, 11
Bangalore, Karnataka 560027
India

Phone: +91 80 4154 1654
Email: psethi@cisco.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: November 7, 2011

R. Singh, Ed.
G. Kalyani
Cisco
Y. Nir
Check Point
Y. Sheffer
Porticor
D. Zhang
Huawei

May 6, 2011

Protocol Support for High Availability of IKEv2/IPsec
draft-ietf-ipsecme-ipsecha-protocol-06

Abstract

The IPsec protocol suite is widely used for business-critical network traffic. In order to make IPsec deployments highly available, more scalable and failure-resistant, they are often implemented as IPsec High Availability (HA) clusters. However there are many issues in IPsec HA clustering, and in particular in IKEv2 clustering. An earlier document, "IPsec Cluster Problem Statement", enumerates the issues encountered in the IKEv2/IPsec HA cluster environment. This document resolves these issues with the least possible change to the protocol.

This document defines an extension to the IKEv2 protocol to solve the main issues of "IPsec Cluster Problem Statement" in the commonly deployed hot-standby cluster, and provides implementation advice for other issues. The main issues solved are the synchronization of IKEv2 Message ID counters, and of IPsec Replay Counters.

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 November 7, 2011.

Copyright Notice

Table of Contents

1. Introduction

The IPsec protocol suite, including IKEv2, is a major building block of virtual private networks (VPNs). In order to make such VPNs highly available, more scalable and failure-resistant, these VPNs are implemented as IKEv2/IPsec Highly Available (HA) clusters. However there are many issues with the IKEv2/IPsec HA cluster. Section 3 and Section 4 below expand on the issues around the IKEv2/IPsec HA cluster solution, issues which were first described in the Problem Statement [4].

In the case of a hot-standby cluster implementation of IKEv2/IPsec based VPNs, the IKEv2/IPsec session is first established between the peer and the active member of the cluster. Later, the active member continuously syncs/updates the IKE/IPsec SA state to the standby member of the cluster. This primary SA state sync-up takes place upon each SA bring-up and/or rekey. Performing the SA state synchronization/update for every single IKE and IPsec message is very costly, so normally it is done periodically. As a result, when the failover event happens, this is first detected by the standby member and, possibly after a considerable amount of time, it becomes the active member. During this failover process the peer is unaware of the failover event, and keeps sending IKE requests and IPsec packets to the cluster, as in fact it is allowed to do because of the IKEv2 windowing feature. After the newly-active member starts, it detects the mismatch in IKE Message ID values and IPsec replay counters and needs to resolve this situation. Please see Section 4 for more details of the problem.

This document defines an extension to the IKEv2 protocol to solve the main issues of IKE Message ID synchronization and IPsec SA replay counter synchronization, and gives implementation advice to address other issues. Following is a summary of the solutions provided in this document:

- IKEv2 Message ID synchronization: this is done by syncing up the expected send and receive Message ID values with the peer, and updating the values at the newly active cluster member.
- IPsec Replay Counter synchronization: this is done by incrementing the cluster’s outgoing SA replay counter values by a "large" number; in addition, the newly-active member requests the peer to increment the replay counter values it is using for the peer’s outgoing traffic.

Although this document describes the IKEv2 Message ID and IPsec replay counter synchronization in the context of an IPsec HA cluster, the solution provided is generic and can be used in other scenarios where IKEv2 Message ID or IPsec SA replay counter synchronization may

be required.

Implementations differ on the need to synchronize the IKEv2 Message ID and/or IPsec replay counters. Both of these problems are handled separately, using a separate notification for each capability. This provides the flexibility of implementing either or both of these solutions.

2. Terminology

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 [1].

"SA Counter Synchronization Request/Response" are the request viz. response of the informational exchange defined in this document to synchronize the IKEv2/IPsec SA counter information between one member of the cluster and the peer.

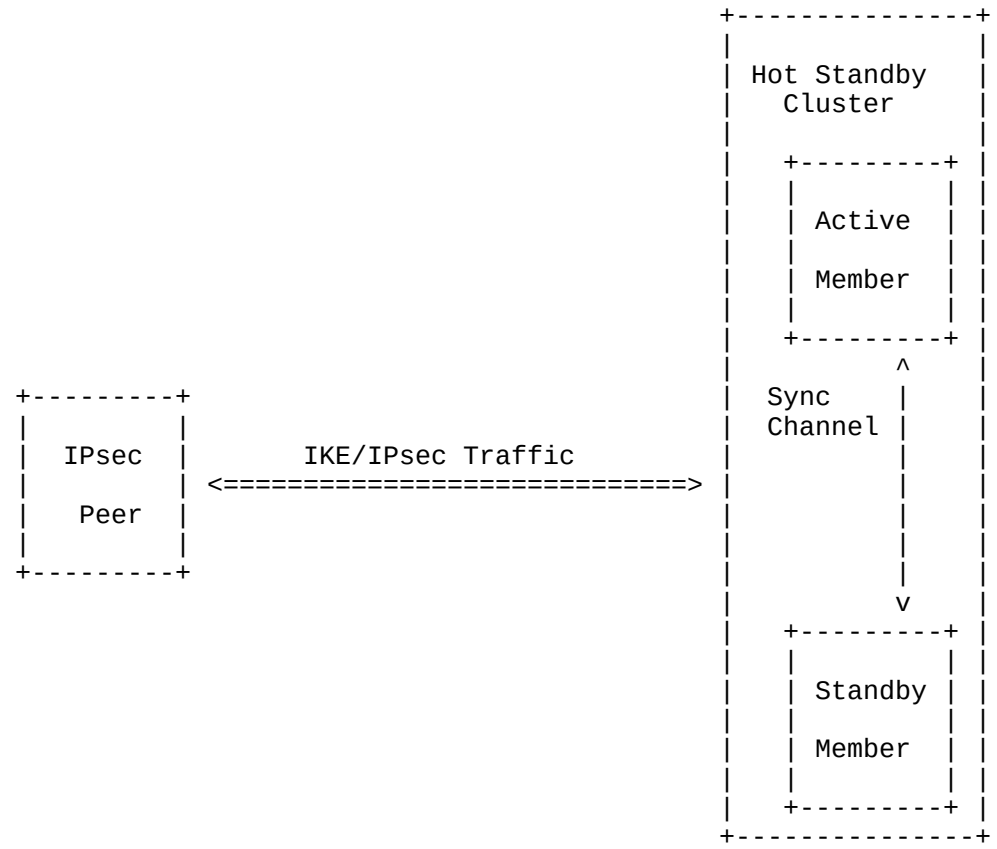
Some of the terms listed below are reused from [4] with further clarification in the context of the current document.

- "Hot Standby Cluster", or "HS Cluster" is a cluster where only one of the members is active at any one time. This member is also referred to as the "active" member, whereas the other(s) are referred to as "standby" members. VRRP [5] is one method of building such a cluster. The goal of the Hot Standby Cluster is to create the illusion of a single virtual gateway to the peer(s).
- "Active Member" is the primary member in the Hot-Standby cluster. It is responsible for forwarding packets on behalf of the virtual gateway.
- "Standby Member" is the primary backup member. This member takes control, i.e. becomes the active member, after the failover event.
- "Peer" is an IKEv2/IPsec endpoint that maintains an IPsec connection with the Hot-Standby cluster. The Peer identifies the cluster by the cluster’s (single) IP address. If a failover event occurs, the standby member of the cluster becomes active, and the peer normally doesn’t notice that failover has taken place. Although we treat the peer as a single entity, it may also be a cluster.
- "Multiple failover" is the situation where, in a cluster with three or more members, multiple failover events happen in rapid succession, e.g. from M1 to M2, and then to M3. It is our goal that the implementation should be able to handle this situation, i.e. to handle the new failover event even if it is still processing the old failover.

o "Simultaneous failover" is the situation where two clusters have an IPsec connection between them, and failover happens at both ends at the same time. It is our goal that implementations should be able to handle simultaneous failover.

The generic term "IKEv2/IPsec SA Counters" is used throughout this document. This term refers to both IKEv2 Message ID counters and IPsec replay counters. According to the IPsec standards, the IKEv2 Message ID counter is mandatory, and used to ensure reliable delivery as well as to protect against message replay in IKEv2; the IPsec SA replay counters are optional, and are used to provide the IPsec anti-replay feature.

Some of these terms are used in the following architectural diagram.



An IPsec Hot Standby Cluster

3. Issues Resolved from IPsec Cluster Problem Statement

The IPsec Cluster Problem Statement [4] enumerates the problems raised by IPsec clusters. The following table lists the problem statement's sections that are resolved by this document.

- o 3.2. Lots of Long Lived State
- o 3.3. IKE Counters
- o 3.4. Outbound SA Counters
- o 3.5. Inbound SA Counters
- o 3.6. Missing Synchronization Messages
- o 3.7. Simultaneous use of IKE and IPsec SAs by Different Members
  - * 3.7.1. Outbound SAs using counter modes
- o 3.8. Different IP addresses for IKE and IPsec
- o 3.9. Allocation of SPIs

The main problem areas are solved using the protocol extension defined below, starting with Section 5; additionally, this section provides implementation advice for other issues in the following subsections. Implementers should note that these subsections include a number of new security-critical requirements.

3.1. Large Amount of State

Section 3.2 of the Problem Statement mentions that a lot of state needs to be synchronized for a cluster to be transparent. The actual volume of that data is very much implementation-dependent, and even for the same implementation, the amounts of data may vary wildly. An IPsec gateway used for inter-domain VPN with a dozen other gateways, and having SAs that are rekeyed every 8 hours, will need a lot less synchronization traffic than a similar gateway used for remote access, and supporting 10,000 clients. This is because counter synchronization is proportional to the number of SAs and requires little data, and the setting up of an SA requires a lot of data. Additionally, remote access IKE and IPsec SA setup tend to happen at a particular time of day, so the example gateway with the 10,000 clients may see 30-50 IKE SA setups per second at 9:00 AM. This would require very heavy synchronization traffic over that short period of time.

If a large volume of traffic is necessary, it may be advisable to use a dedicated high-speed network interface for synch traffic. When packet loss can be made extremely low, it may be advisable to use a stateless transport such as UDP, to minimize network overhead.

If these methods are insufficient, it may be prudent that for some SAs the entire state is not synchronized. Instead, only an indication of the SA's existence is synchronized. This, in combination with a sticky solution (as described in section 3.7 of

the problem statement) ensures that the traffic from a particular peer does not reach a different member before an actual failover happens. When that happens, the method described in [6] can be used to quickly force the peer to set up a new SA.

3.2. Multiple Members Using the Same SA

In a load-sharing cluster of the "duplicate" variety (see section 3.7 of the problem statement) multiple members may need to send traffic with the same selectors. To actually use the same SA the cluster would have to synchronize the Replay Counter after every packet, and that would impose unreasonable requirements on the synch connection.

A far better solution would be to not synchronize the outbound SA, and create multiple outbound SAs, one for each member. The problem with this option is that the peer might view these multiple parallel SAs as redundant, and tear down all but one of them.

Section 2.8 of [2] specifically allows multiple parallel SAs, but the reason given for this is to have multiple SAs with different QoS attributes. So while this is not a new requirement of IKEv2 implementations working with QoS, we re-iterate here that IPsec peers MUST accept the long-term existence of multiple parallel SAs, even when QoS mechanisms are not in use.

3.3. Avoiding Collisions in SPI Number Allocation

Section 3.9 of the problem statement describes the problem of two cluster members allocating the same SPI number for two different SAs. This would violate section 4.4.2.1 of [3]. There are several schemes to allow implementations to avoid such collisions, such as partitioning the SPI space, a request-response over the synch channel, and locking mechanisms. We believe that these are sufficiently robust and available so that we don’t need to make an exception to RFC 4301, and we can leave this problem for the implementations to solve. Cluster members must not generate multiple inbound SAs with the same SPI.

3.4. Interaction with Counter Modes

For SAs involving counter mode ciphers such as CTR [7] or GCM [8] there is yet another complication. The initial vector for such modes MUST NOT be repeated, and senders may use methods such as counters or LFSRs to ensure this property. For an SA shared between multiple active members (load sharing cases), implementations MUST ensure that no initial vector is ever repeated. Similar concerns apply to an SA failing over from one member to another. See [9] for a discussion of this problem in another context.

Just as in the SPI collision problem, there are ways to avoid a collision of initial vectors, and this is left up to implementations. In the context of load sharing, parallel SAs are a simple solution to this problem as well.

4. The IKEv2/IPsec SA Counter Synchronization Problem

The IKEv2 protocol [2] states that "An IKE endpoint MUST NOT exceed the peer's stated window size for transmitted IKE requests".

All IKEv2 messages are required to follow a request-response paradigm. The initiator of an IKEv2 request MUST retransmit the request, until it has received a response from the peer. IKEv2 introduces a windowing mechanism that allows multiple requests to be outstanding at a given point of time, but mandates that the sender's window should not move until the oldest message it has sent is acknowledged. Loss of even a single message leads to repeated retransmissions followed by an IKEv2 SA teardown if the retransmissions remain unacknowledged.

An IPsec Hot Standby Cluster is required to ensure that in the case of failover, the standby member becomes active immediately. The standby member is expected to have the exact value of the Message ID counter as the active member had before failover. Even assuming the best effort to update the Message ID values from active to standby member, the values at the standby member can still be stale due to the following reasons:

- The standby member is unaware of the last message that was received and acknowledged by the previously active member, as the failover event could have happened before the standby member could be updated.
- The standby member does not have information about on-going unacknowledged requests sent by the previously active member. As a result after the failover event, the newly active member cannot retransmit those requests.

When a standby member takes over as the active member, it can only initialize the Message ID values from the previously updated values. This would make it reject requests from the peer when these values are stale. Conversely, the standby member may end up reusing a stale Message ID value which would cause the peer to drop the request. Eventually there is a high probability of the IKEv2 and corresponding IPsec SAs getting torn down simply because of a transitory Message ID mismatch and retransmission of requests, negating the benefits of the high availability cluster despite the periodic update between the cluster members.

A similar issue is also observed with IPsec anti-replay counters if anti-replay protection is enabled, which is commonly the case. Regardless of how well the ESP and AH SA counters are synchronized from the active to the standby member, there is a chance that the standby member would end up with stale counter values. The standby member would then use those stale counter values when sending IPsec packets. The peer would drop such packets since when the anti-replay protection feature is enabled, duplicate use of counters is not allowed. Note that IPsec allows the sender to skip some counter values and continue sending with higher counter values.

We conclude that a mechanism is required to ensure that the standby member has correct Message ID and IPsec counter values when it becomes active, so that sessions are not torn down as a result of mismatched counters.

5. SA Counter Synchronization Solution

This document defines two separate approaches to resolving the issues of mismatched IKE Message ID values and IPsec counter values.

- In the case of IKE Message ID values, the newly active cluster member and the peer negotiate a pair of new values so that future IKE messages will not be dropped.
- For IPsec counter values, the newly-active member and the peer both increment their respective counter values, "skipping forward" by a large number, to ensure that no IPsec counters are ever reused.

Although conceptually separate, the two synchronization processes would typically take place simultaneously.

First, the peer and the active member of the cluster negotiate their ability to support IKEv2 Message ID synchronization and/or IPsec Replay Counter synchronization. This is done by exchanging one or both of the IKEV2_MESSAGE_ID_SYNC_SUPPORTED and IPSEC_REPLAY_COUNTER_SYNC_SUPPORTED notifications during the IKE_AUTH exchange. When negotiating these capabilities, the responder MUST NOT assert support of a capability unless such support was asserted by the initiator. Only a capability whose support was asserted by both parties can be used during the lifetime of the SA. The peer’s capabilities with regard to this extension are part of the IKEv2 SA state, and thus MUST be shared between the cluster members.

This per-IKE SA information is shared with the other cluster members.

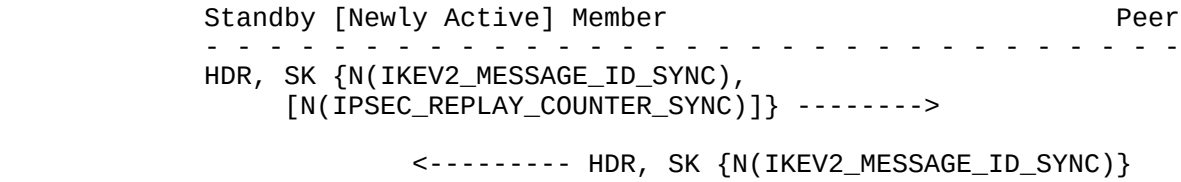
```
Peer                                                    Active Member
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
HDR, SK {IDi, [CERT], [CERTREQ], [IDr], AUTH,
     [N(IKEV2_MESSAGE_ID_SYNC_SUPPORTED),]
     [N(IPSEC_REPLAY_COUNTER_SYNC_SUPPORTED),]
     SAi2, TSi, TSr} ---------->

<-------- HDR, SK {IDr, [CERT+], [CERTREQ+], AUTH,
               [N(IKEV2_MESSAGE_ID_SYNC_SUPPORTED),]
               [N(IPSEC_REPLAY_COUNTER_SYNC_SUPPORTED),] SAr2, TSi, TSr}
```

After a failover event, the standby member MAY use the IKE Message ID and/or IPsec Replay Counter synchronization capability when it becomes the active member, and provided support for the capabilities used has been negotiated. Following that, the peer MUST respond to any synchronization message it receives from the newly-active cluster member, subject to the rules noted below.

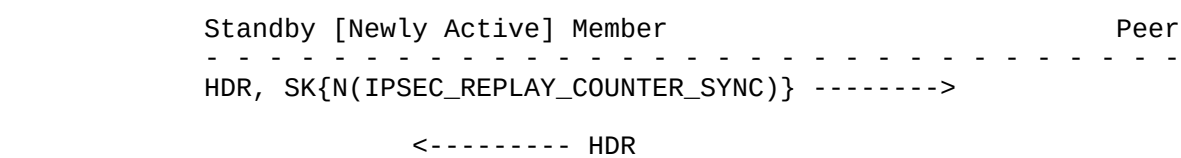
After the failover event, when the standby member becomes active, it has to synchronize its SA counters with the peer. There are now four possible cases:

1. The cluster member wishes to only perform IKE Message ID value synchronization. In this case it initiates an Informational exchange, with Message ID zero and the sole notification IKEV2_MESSAGE_ID_SYNC.
2. If the newly-active member wishes to perform only IPsec replay counter synchronization, it generates a regular IKEv2 Informational exchange using the current Message ID values, and containing the IPSEC_REPLAY_COUNTER_SYNC notification.
3. If synchronization of both counters is needed, the cluster member generates a zero-Message ID message as in case #1, and includes both notifications in this message.
4. Lastly, the peer may not support this extension. This is known to the newly-active member (because the cluster members must share this information, as noted earlier). This case is the existing IKEv2 behavior, and the IKE and IPsec SAs may or may not survive the failover, depending on the exact state on the peer and the cluster member.

This figure contains the IKE message exchange used for SA counter synchronization. The following subsections describe the details of the sender and receiver processing of each message.



Alternatively, if only IPsec Replay Counter synchronization is desired, a normal Informational exchange is used, where the Message ID is non-zero:



5.1. Processing Rules for IKE Message ID Synchronization

The newly-active member sends a request containing two counter values, one for the member (itself) and another for the peer, as well as a random nonce. We denote the values M1 and P1. The peer responds with a message containing two counter values, M2 and P2 (note that the values appear in the opposite order in the notification's payload). The goal of the rules below is to prevent an attacker from replaying a synchronization message, thereby invalidating IKE messages that are currently in process.

- M1 is the next sender's Message ID to be used by the member. M1 MUST be chosen so that it is larger than any value known to have been used. It is RECOMMENDED to increment the known value at least by the size of the IKE sender window.
- P1 SHOULD be 1 more than the last Message ID value received from the peer, but may be any higher value.
- The member SHOULD communicate the sent values to the other cluster members, so that if a second failover event takes place, the synchronization message is not replayed. Such a replay would result in the eventual deletion of the IKE SA (see below).
- The peer MUST silently drop any received synchronization message if M1 is lower than or equal to the highest value it has seen from the cluster. This includes any previous received synchronization messages.
- M2 MUST be at least the higher of the received M1, and one more than the highest sender value received from the cluster. This includes any previous received synchronization messages.

- o P2 MUST be the higher of the received P1 value, and one more than the highest sender value used by the peer.
- o The request contains a Nonce field. This field MUST be returned in the response, unchanged. A response MUST be silently dropped if the received Nonce does not match the one that was sent.
- o Both the request and the response MUST NOT contain any additional payloads, other than an optional IPSEC_REPLAY_COUNTER_SYNC notification in the request.
- o The request and the response MUST both be sent with a Message ID value of zero.

5.2. Processing Rules for IPsec Replay Counter Synchronization

Upon failover, the newly-active member MUST increment its own Replay Counter (the counter used for outgoing traffic), so as to prevent the case of its traffic being dropped by the peer as replay. We note that IPsec allows the replay counter to skip forward by any amount. The estimate is based on the outgoing IPsec bandwidth and the frequency of synchronization between cluster members. In those implementations where it is difficult to estimate this value, the counter can be incremented by a very large number, e.g. 2**30. In the latter case, a rekey SHOULD follow shortly afterwards, to ensure that the counter never wraps around.

Next, the cluster member estimates the number of incoming messages it might have missed, using similar logic. The member sends out a IPSEC_REPLAY_COUNTER_SYNC notification, either stand-alone or together with a IKEV2_MESSAGE_ID_SYNC notification.

If the IPSEC_REPLAY_COUNTER_SYNC is included in the same message as IKEV2_MESSAGE_ID_SYNC, the peer MUST process the Message ID notification first (which might cause the entire message to be dropped as a replay). Then, it MUST increment the replay counters for all Child SAs associated with the current IKE SA by the amount requested by the cluster member.

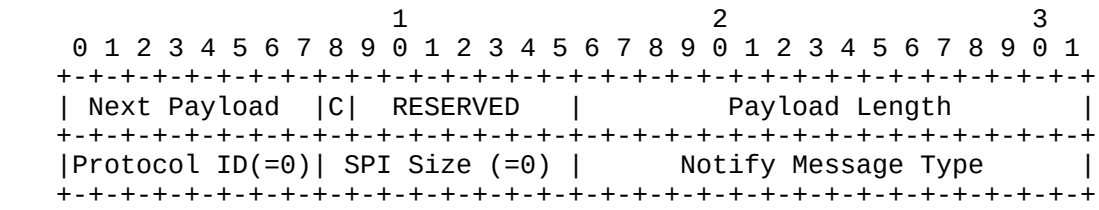
6. IKEv2/IPsec Synchronization Notification Payloads

This section lists the new notification payload types defined by this extension.

All multi-octet fields representing integers are laid out in big endian order (also known as "most significant byte first", or "network byte order").

6.1. The IKEV2_MESSAGE_ID_SYNC_SUPPORTED Notification

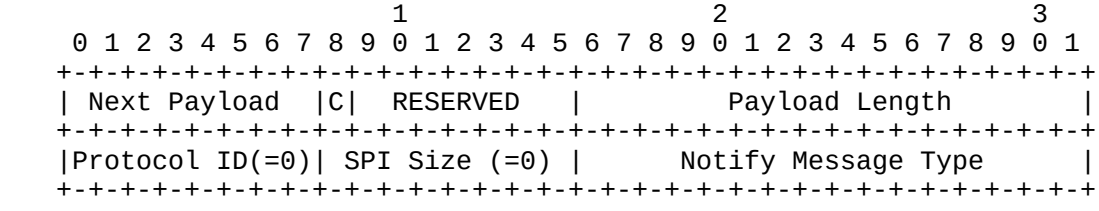
This notification payload is included in the IKE_AUTH request/ response to indicate support of the IKEv2 Message ID synchronization mechanism described in this document.



The 'Next Payload', 'Payload Length', 'Protocol ID', 'SPI Size', and 'Notify Message Type' fields are the same as described in Section 3 of [2]. The 'SPI Size' field MUST be set to 0 to indicate that the SPI is not present in this message. The 'Protocol ID' MUST be set to 0, since the notification is not specific to a particular security association. The 'Payload Length' field is set to the length in octets of the entire payload, including the generic payload header. The 'Notify Message Type' field is set to indicate IKEV2_MESSAGE_ID_SYNC_SUPPORTED, value TBD by IANA. There is no data associated with this notification.

6.2. The IPSEC_REPLAY_COUNTER_SYNC_SUPPORTED Notification

This notification payload is included in the IKE_AUTH request/ response to indicate support for the IPsec SA Replay Counter synchronization mechanism described in this document.

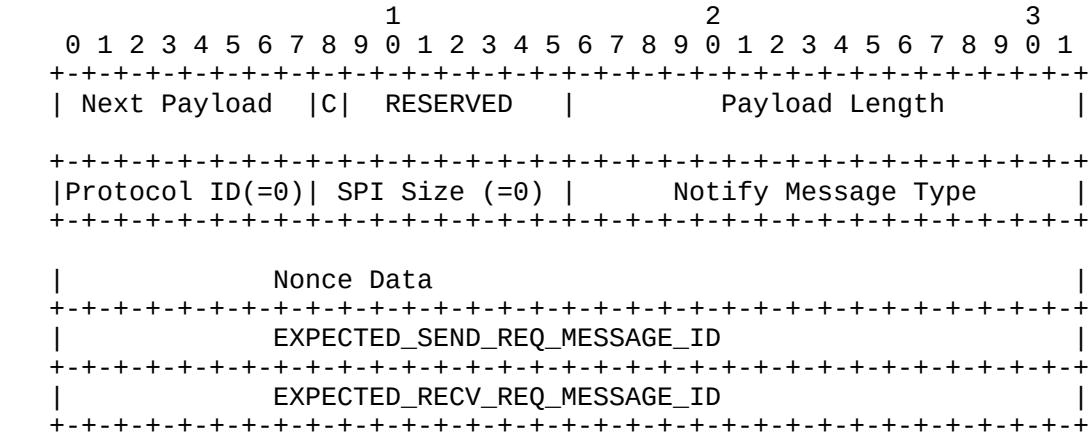


The 'Next Payload', 'Payload Length', 'Protocol ID', 'SPI Size', and 'Notify Message Type' fields are the same as described in Section 3 of [2] . The 'SPI Size' field MUST be set to 0 to indicate that the SPI is not present in this message. The 'Protocol ID' MUST be set to 0, since the notification is not specific to a particular security association. The 'Payload Length' field is set to the length in octets of the entire payload, including the generic payload header. The 'Notify Message Type' field is set to indicate IPSEC_REPLAY_COUNTER_SYNC_SUPPORTED, value TBD by IANA. There is no

data associated with this notification.

6.3. The IKEV2_MESSAGE_ID_SYNC Notification

This notification payload type (value TBD by IANA) is defined to synchronize the IKEv2 Message ID values between the newly-active (formerly standby) cluster member and the peer.



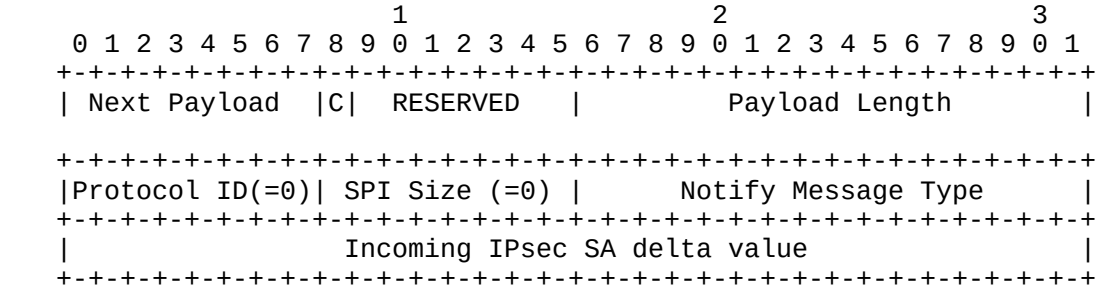
It contains the following data.

- Nonce Data (4 octets): the random nonce data. The data should be identical in the synchronization request and response.
- EXPECTED_SEND_REQ_MESSAGE_ID (4 octets): this field is used by the sender of this notification payload to indicate the Message ID it will use in the next request that it will send to the other protocol peer.
- EXPECTED_RECV_REQ_MESSAGE_ID (4 octets): this field is used by the sender of this notification payload to indicate the Message ID it is expecting in the next request to be received from the other protocol peer.

6.4. The IPSEC_REPLAY_COUNTER_SYNC Notification

This notification payload type (value TBD by IANA) is defined to synchronize the IPsec SA Replay Counters between the newly-active (formerly standby) cluster member and the peer. Since there may be numerous IPsec SAs established under a single IKE SA, we do not directly synchronize the value of each one. Instead, a delta value is sent and all Replay Counters for Child SAs of this IKE SA are incremented by the same value. Note that this solution requires that either all Child SAs use Extended Sequence Numbers or else that no

Child SA uses Extended Sequence Numbers [3]. This notification is only sent by the cluster.



The notification payload contains the following data.

- Incoming IPsec SA delta value (4 or 8 octets): The sender requests that the peer should increment all the Child SA Replay Counters for the sender's incoming (the peer's outgoing) traffic by this value. The size of this field depends on the ESN bit associated with the Child SAs: if the ESN bit is 1, the field's size is 8 octets, otherwise it is 4 octets. We note that this constrains the Child SAs of each IKE SA to either all have the ESN bit on or off.

7. Implementation Details

This protocol does not change any of the existing IKEv2 rules regarding Message ID values.

The standby member can initiate the synchronization of IKEv2 Message ID's under different circumstances.

- When it receives a problematic IKEv2/IPsec packet, i.e. a packet outside its expected receive window.
- When it has to send the first IKEv2/IPsec packet after a failover event.
- When it has just received control from the active member and wishes to update the values proactively, so that it need not start this exchange later, when sending or receiving the request.

To clarify the first alternative: the normal IKE behavior of rejecting out-of-window messages is not changed, but such messages can still be a valid trigger for the exchange defined in this document. To avoid DoS attacks resulting from replayed messages, the peer MUST NOT initiate counter synchronization for any particular IKE SA more than once per failover event.

The standby member can initiate the synchronization of IPsec SA

Replay Counters:

- If there has been traffic using the IPsec SA in the recent past and the standby member suspects that its Replay Counter may be stale.

Since there can be a large number of sessions at the standby member, and sending synchronization exchanges for all of them may result in overload, the standby member can choose to initiate the exchange in a "lazy" fashion: only when it has to send or expects to receive traffic from each peer. In general, the standby member is free to initiate this exchange at its discretion. Implementation considerations include the ability to survive a certain amount of traffic loss, and the capacity of a cluster member to initiate counter synchronization simultaneously with a large number of peers.

8. IKE SA and IPsec SA Message Sequencing

The straightforward definitions of message sequence numbers, retransmissions and replay protection in IPsec and IKEv2 are strained by the failover scenarios described in this document. This section describes some policy choices that need to be made by implementations in this setting.

8.1. Handling of Pending IKE Messages

After sending its "receive" counter, the cluster member MUST reject (silently drop) any incoming IKE messages that are outside its declared window. A similar rule applies to the peer. Local policies vary, and strict implementations will reject any incoming IKE message arriving before Message ID synchronization is complete.

8.2. Handling of Pending IPsec Messages

For IPsec, there is often a trade-off between security and reliability of the protected protocols. Here again there is some leeway for local policy. Some implementations might accept incoming traffic that is outside the replay window for some time after the failover event, and until the counters had been synchronized. Strict implementations will only accept traffic that’s inside the "safe" window.

8.3. IKE SA Inconsistencies

IKEv2 is normally a reliable protocol. As long as an IKE SA is valid, both peers share a single, consistent view of the IKE SA and all associated Child SAs. Failover situations as described in this document may involve forced deletion of IKE messages, resulting in

inconsistencies, such as Child SAs that exist on only one of the peers. Such SAs might cause an INVALID_SPI to be returned when used by that peer. Note that Sec. 1.5 of [2] allows but does not mandate sending an INVALID_SPI notification in this case.

The Working Group discussed at some point a proposed set of rules for dealing with such situations. However we believe that these situations should be rare in practice; as a result the "default" behavior of tearing down the entire IKE SA is to be preferred over the complexity of dealing with a multitude of edge cases.

9. Step by Step Details

This section goes through the sequence of steps of a typical failover event, looking at a case where the IKEv2 Message ID values are synchronized.

- The active cluster member and the peer device establish the session. They both announce the capability to synchronize counter information by sending the IKEV2_MESSAGE_ID_SYNC_SUPPORTED notification in the IKE_AUTH Exchange.
- Some time later, the active member dies, and a standby member takes over. The standby member sends its own idea of the IKE Message IDs (both incoming and outgoing) to the peer in an Informational message exchange with Message ID zero.
- The peer first authenticates the message. The peer compares the received values with the values available locally and picks the higher value. It then updates its Message IDs with the higher values and also propose the same values in its response.
- The peer should not wait for any pending responses while responding with the new Message ID values. For example, if the window size is 5 and the peer’s window is 3-7, and if the peer has sent requests 3, 4, 5, 6, 7 and received responses only for 4, 5, 6, 7 but not for 3, then it should include the value 8 in its EXPECTED_SEND_REQ_MESSAGE_ID payload and should not wait for a response to message 3 anymore.
- Similarly, the peer should also not wait for pending (incoming) requests. For example if the window size is 5 and the peer’s window is 3-7 and if the peer has received requests 4, 5, 6, 7 but not 3, then it should send the value 8 in the EXPECTED_RECV_REQ_MESSAGE_ID payload, and should not expect to receive message 3 anymore.

10. Interaction with other specifications

The usage scenario of this IKEv2/IPsec SA counter synchronization solution is that an IKEv2 SA has been established between the active

member of a hot-standby cluster and a peer, followed by a failover event occurring and the standby member becoming active. The solution further assumes that the IKEv2 SA state was continuously synchronized between the active and standby members of the cluster before the failover event.

- Session resumption [10] assumes that a peer (client or initiator) detects the need to re-establish the session. In IKEv2/IPsec SA counter synchronization, it is the newly-active member (a gateway or responder) that detects the need to synchronize the SA counter after the failover event. Also in a hot-standby cluster, the peer establishes the IKEv2/IPsec session with a single IP address that represents the whole cluster, so the peer normally does not detect the event of failover in the cluster unless the standby member takes too long to become active and the IKEv2 SA times out by use of the IKEv2 liveness check mechanism. To conclude, session resumption and SA counter synchronization after failover are mutually exclusive: they are not expected to be used together, and both features can coexist within the same implementation without affecting each other.
- The IKEv2 Redirect mechanism for load-balancing [11] can be used either during the initial stages of SA setup (the IKE_SA_INIT and IKE_AUTH exchanges) or after session establishment. SA counter synchronization is only useful after the IKE SA has been established and a failover event has occurred. So, unlike Redirect, it is irrelevant during the first two exchanges. Redirect after the session has been established is mostly useful for timed or planned shutdown/maintenance. A real failover event cannot be detected by the active member ahead of time, and so using Redirect after session establishment is not possible in the case of failover. So, Redirect and SA counter synchronization after failover are mutually exclusive, in the sense described above.
- IKEv2 Failure Detection [6] solves a similar problem where the peer can rapidly detect that a cluster member has crashed based on a token. It is unrelated to the current scenario because the goal in failover is for the peer not to notice that a failure has occurred.

11. Security Considerations

Since Message ID synchronization messages need to be sent with Message ID zero, they are potentially vulnerable to replay attacks. Because of the semantics of this protocol, these can only be denial-of-service (DoS) attacks, and we are aware of two variants.

- Replay of Message ID synchronization request: This is countered by the requirement that the Send counter sent by the cluster member should always be monotonically increasing, a rule that the peer

enforces by silently dropping messages that contradict it.

- o Replay of the Message ID synchronization response: This is countered by sending the nonce data along with the synchronization payload. The same nonce data has to be returned in the response. Thus the standby member will accept a reply only for the current request. After it receives a valid response, it MUST NOT process the same response again and MUST discard any additional responses.

As mentioned in Section 7, trigerring counter synchronization by out-of-window, potentially replayed messages, could open a DoS vulnerability. This risk is mitigated by the solution described in that section.

12. IANA Considerations

This document introduces four new IKEv2 Notification Message types as described in Section 6. The new Notify Message Types must be assigned values between 16396 and 40959.

Name	Value
IKEV2_MESSAGE_ID_SYNC_SUPPORTED	TBD by IANA
IPSEC_REPLAY_COUNTER_SYNC_SUPPORTED	TBD by IANA
IKEV2_MESSAGE_ID_SYNC	TBD by IANA
IPSEC_REPLAY_COUNTER_SYNC	TBD by IANA

13. Acknowledgements

We would like to thank Pratima Sethi and Frederic Detienne for their review comments and valuable suggestions for the initial version of the document.

We would also like to thank the following people (in alphabetical order) for their review comments and valuable suggestions: Dan Harkins, Paul Hoffman, Steve Kent, Tero Kivinen, David McGrew, and Pekka Riikonen.

14. Change Log

This section lists all the changes in this document.

NOTE TO RFC EDITOR: Please remove this section before publication.

14.1. Draft -06

Applied multiple review comments, from Pekka Riikonen, Alexey Melnikov, Stephen Farrel, Robert Sparks, Pete Resnick, Russ Housley and Adrian Farrel. Added an architectural reference diagram. Added a MUST requirement for cluster members to share peers' support of this protocol, which had been implicit in previous versions.

14.2. Draft -05

Applied Sean Turner's review comments.

14.3. Draft -04

Extended Sec. 3 for better coverage of other IPsec cluster-related issues, and how they are resolved within the existing standards.

14.4. Draft -03

Clarified the rules for Message ID sync, so that replay attacks can be avoided without a failover counter.

Added wording regarding inconsistent IKE state (basically choosing to ignore the problem) and further rules dealing with pending traffic.

The IPsec replay counter delta value now refers to incoming traffic. The associated notification is only sent from the cluster to the peer, and not back.

14.5. Draft -02

Addressed comments by Yaron Sheffer posted on the WG mailing list.

Numerous editorial changes.

14.6. Draft -01

Added "Multiple and Simultaneous failover" scenarios as pointed out by Pekka Riikonen.

Now document provides a mechanism to sync either IKEv2 message or IPsec replay counter or both to cater different types of implementations.

HA cluster's "failover count' is used to encounter replay of sync requests by attacker.

The sync of IPsec SA replay counter optimized to to have just one

global bumped-up outgoing IPsec SA counter of ALL Child SAs under an IKEv2 SA.

The examples added for IKEv2 Message ID sync to provide more clarity.

Some edits as per comments on mailing list to enhance clarity.

14.7. Draft -00

Version 00 is identical to draft-kagarigi-ipsecme-ikev2-windowsync-04, started as WG document.

Added IPSECME WG HA design team members as authors.

Added comment in Introduction to discuss the window sync process on WG mailing list to solve some concerns.

15. References

15.1. Normative References

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

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

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

15.2. Informative References

[4] Nir, Y., "IPsec Cluster Problem Statement", RFC 6027, October 2010.

[5] Nadas, S., "Virtual Router Redundancy Protocol (VRRP) Version 3 for IPv4 and IPv6", RFC 5798, March 2010.

[6] Nir, Y., Wierbowski, D., Detienne, F., and P. Sethi, "A Quick Crash Detection Method for IKE", draft-ietf-ipsecme-failure-detection-08 (work in progress), April 2011.

[7] Housley, R., "Using Advanced Encryption Standard (AES) Counter Mode With IPsec Encapsulating Security Payload (ESP)", RFC 3686, January 2004.

[8] Viega, J. and D. McGrew, "The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP)", RFC 4106, June 2005.

[9] McGrew, D. and B. Weis, "Using Counter Modes with Encapsulating Security Payload (ESP) and Authentication Header (AH) to Protect Group Traffic", RFC 6054, November 2010.

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

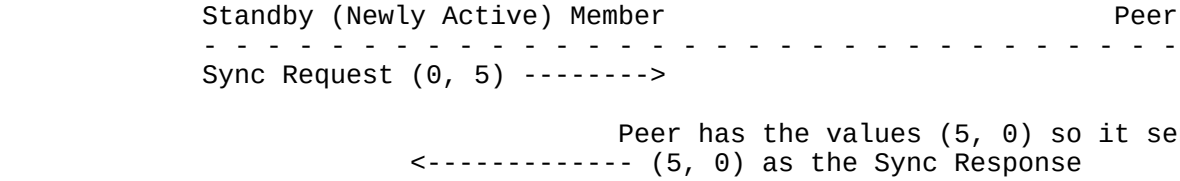
[11] Devarapalli, V. and K. Weniger, "Redirect Mechanism for the Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 5685, November 2009.

Appendix A. IKEv2 Message ID Sync Examples

This (non-normative) section presents some examples that illustrate how the IKEv2 Message ID values are synchronized. We use a tuple notation, denoting the two counters EXPECTED_SEND_REQ_MESSAGE_ID and EXPECTED_RECV_REQ_MESSAGE_ID on each protocol party as (EXPECTED_SEND_REQ_MESSAGE_ID, EXPECTED_RECV_REQ_MESSAGE_ID).

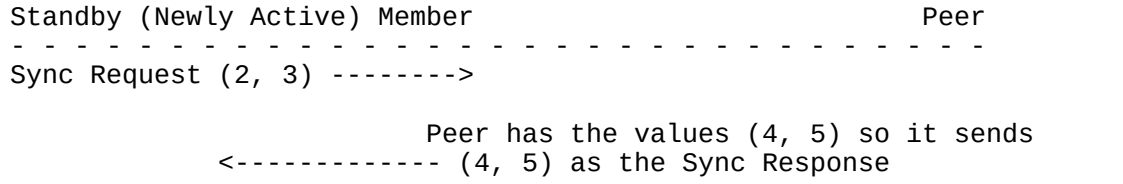
Note that if the IKE message counters are already synchronized (as in the first example), we expect the numbers to be reversed between the two sides. If one protocol party intends to send the next request as 4, then the other expects the next received request to be 4.

A.1. Normal Failover - Example 1



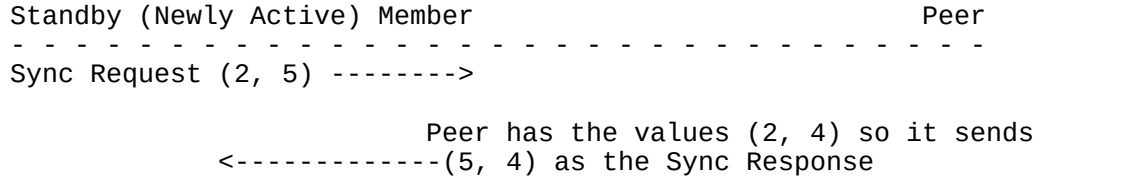
In this example, the peer has most recently sent an IKE request with Message ID 4, and has never received a request. So the peer's expected values for the next pair of messages are (5, 0). These are the same values as received from the member and therefore they are sent as-is.

A.2. Normal Failover - Example 2



In this example, the peer has most recently sent an IKE message with the Message ID 3, and received one with ID 4. So the peer's expected values for the next pair of messages are (4, 5). These are both higher than the corresponding values just received from the member (the order of tuple members is reversed when doing this comparison!), and therefore they are sent as-is.

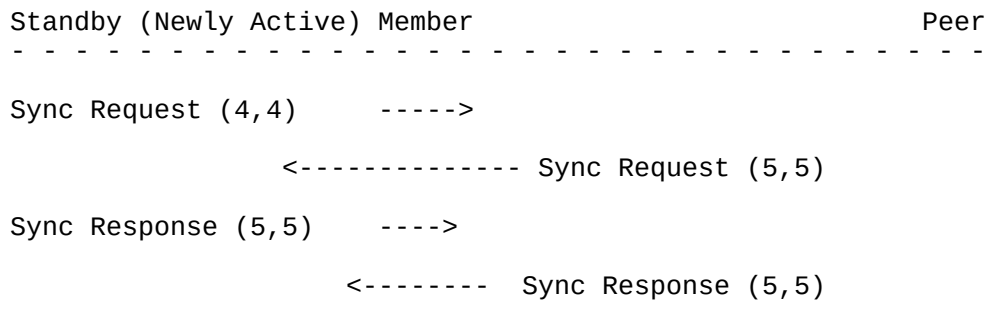
A.3. Normal Failover - Example 3



In this example, the newly active member expects to send the next IKE message with ID 2. It sends an expected receive value of 5, which is higher than the last ID value it has seen from the peer, because it believes some incoming messages may have been lost. The peer has last sent a message with ID 1, and received one with ID 3, indicating that the a couple of messages sent by the previously active member had not been synchronized into the other member. So the peer's next expected (send, receive) values are (2, 4). The peer replies with the maximum of the received and the expected value for both send and receive counters: (max(2, 5), max(4, 2)) = (5, 4).

A.4. Simultaneous Failover

In the case of simultaneous failover, both sides send their synchronization requests simultaneously. The eventual outcome of synchronization consists of the higher counter values. This is demonstrated in the following figure.



Authors' Addresses

Raj Singh (Editor)
Cisco Systems, Inc.
Divyashree Chambers, B Wing, O'Shaugnessy Road
Bangalore, Karnataka 560025
India

Phone: +91 80 4301 3320
Email: rsj@cisco.com

Kalyani Garigipati
Cisco Systems, Inc.
Divyashree Chambers, B Wing, O'Shaugnessy Road
Bangalore, Karnataka 560025
India

Phone: +91 80 4426 4831
Email: kagarigi@cisco.com

Yoav Nir
Check Point Software Technologies Ltd.
5 Hasolelim St.
Tel Aviv 67897
Israel

Email: ynir@checkpoint.com

Yaron Sheffer
Porticor Cloud Security

Email: yaronf.ietf@gmail.com

Dacheng Zhang
Huawei Technologies Ltd.

Email: zhangdacheng@huawei.com