IESG   F. Detienne
Internet-Draft   P. Sethi
Expires: January 30, 2010   Cisco
                                                         Y. Nir
                                                    Check Point
                                                  July 29, 2009

                         Safe IKE Recovery
                    draft-detienne-ikev2-recovery-03

Status of this Memo

   Internet-Drafts are working documents of the Internet Engineering
   Task Force (IETF), its areas, and its working groups. Note that
   other groups may also distribute working documents as Internet-
   Drafts.

   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."

   The list of current Internet-Drafts can be accessed at
   http://www.ietf.org/ietf/1id-abstracts.txt.

   The list of Internet-Draft Shadow Directories can be accessed at
   http://www.ietf.org/shadow.html.

   This Internet-Draft will expire on January 30, 2010.

Copyright Notice

Abstract

The Internet Key Exchange protocol version 2 (IKEv2) suffers from the
limitation of not having a means to quickly recover from a stale
state known as dangling Security Associations (SA's) where one side
has SA's that the corresponding party does not have anymore.

This Draft proposes to address the limitation by offering an
immediate, DoS-free recovery mechanism for IKE that can be used in
all failover or post-crash situations.

Table of Contents

1. Introduction

   If an IKEv2 ([IKEv2]) endpoint receives an IPsec packet that it does
   not recognize (invalid SPI), a specific notify (INVALID_SPI) can be
   sent back to the originating peer to take action. This payload is
   typically only going to be trusted if it is protected by a IKE_SA as
   unprotected notifies can easily be forged. Similarly, an IKEv2
   endpoint receiving an unrecognized IKE message MAY send back an
   INVALID_IKE_SPI notify to the originating peer. In order to validate
   those unauthenticated messages, a polling sequence has to be started.

   The polling sequence works as follow. When a peer doubts the
   liveness of its remote peer, it can send empty informational
   exchanges expecting a reply confirming liveness. This works as
   informational exchanges are supposed to be acknowledged in IKEv2.

   Practical mechanisms offered so far suffer from one of the following
   limitations:
   o poll based and slow to react or resource hungry
   o based on unauthenticated packets and hence open to denial of
      service attacks
   o resource intensive (mostly CPU)
   This memo proposes to decrease the time incurred by this sequence.

   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 [Bra97].

2. Protocol overview

2.1. Protocol requirements

   Dangling SA's can arise from many situations and in many network
   deployment contexts. The protocol described herein is meant to solve
   the dangling SA problem in all possible contexts, without making any
   assumption on resource availability that IKE does not already make.
   Since speed is the main driver for this memo, the protocol must
   minimize the time taken to identify then repair a dangling SA
   condition.

2.1.1. Security level

   The security level of the protocol is targeted to be at least as
   secure as IKE itself; i.e., the recovery protocol MUST NOT offer an
   entry point to an attack that IKE alone could resist to (sufficient
   level of security).

2.1.2. Network scenarios

2.1.2.1. Remote access vs. lan-to-lan

   The protocol must be independent of the network deployment. I.e.,
   the protocol must be usable in lan-to-lan as well as in remote-access
   types of situations or any other use case that can be deployed. No
   restriction must be made for the scope of the recovery protocol.

2.1.2.2. Failover pairs and clusters

   The protocol must work in the presence of failover pairs or clusters.
   Some network deployments will involve clusters of devices acting as a
   single device and those clusters may be extremely large. No
   assumption must be made as to the devices or group of devices that
   will implement the protocols. No back-channel mechanism should be
   necessary for the cluster to support the recovery protocol fully and
   securely; it is not expected that the cluster will be practically
   able to maintain communication between its elements, yet failover and
   recovery between elements of the cluster must remain possible.

2.1.2.3. Transition smoothness

   The protocol must cover for transitions; in particular, it must cover
   what happens when the SA search indices are changed. For example, if
   the (IKE_SPI,src_addr,dst_addr) tuple changes as would be the case in
   an IKE rekey or a MOBIKE update, the recovery protocol must support
   that transition smoothly.

   Ideally, the protocol must be insensitive to SA index changes and
   must avoid deleting state in order to preserve the data flow until
   new SA's are in place to take care of it.

2.1.3. Lightweightness

   The protocol should be lightweight and consume minimum amounts of
   network, memory and computational resources to validate dangling
   state. Ideally, the protocol should not consume anything that is not
   strictly necessary to the security and validity of the protocol.
   Also, the protocol should avoid state creation until absolutely
   necessary.

   This requirement is particularly important for resource constrained
   devices as well as for multiprotocol devices whose memory and CPU is
   claimed by and shared with other protocols or tasks. This
   requirement is the main reason for ruling out Birth Certificates as
   described in [BIRTHCERT].

2.2. High level description

   The recovery procedure works in 3 stages:
   1. An invalid IKE or ESP packet is received by either peer
   2. The remote peer is notified through a protected or unprotected
       notify
       * Protected notifies are implicitly trusted
       * The remote peer attemps to confirm the legitimacy of
          Unprotected Notifies
   3. The remote peer deletes or recreates the SA's in error

2.3. Notation

   The IKEv2 notation will be used throughout this document with one
   notable addition. Parent SA describes an IKE_SA from which a
   CHILD_SA has been derived.

   The following notation is specific to this document:
   o Cookie_X: a cookie generated by peer X that is to be reflected by
      peer Y.
   o CHECK_SPI(QUERY/ACK/NACK, Cookie): a contextual notation to
      express that cookie data is attached to a CHECK_SPI payload
      CHECK_SPI(QUERY/ACK/NACK). See Section 6.

2.4. Protocol design guidelines

   The general approach to recovering from dangling SA situations is to
   send proofs of desynchronization and liveness. It is admittedly
   difficult for two gateways to demonstrate they did have SA's but have
   lost them without a secure, authenticated channel to do so. It is
   however relatively easy for these gateways to provide valuable hints
   about the lost SA's.

   This memo presents a protocol that builds enough trust for those
   hints to be taken in account. The basic principle is that an
   attacker taking advantage of this recovery procedure would have to be
   positioned on the network such that it could perform more interesting
   attacks than tackling recovery. I.e. the barrier for attacking IKE
   recovery is as high or higher than other parts of the IKE protocol.

   The recovery of SA's as outlined in this memo occurs in three phases:
   o Unrecognized SPI's are detected
   o The protocol collects clues of previous connectivity
   o The SA's are repaired by [IKEv2] or by reconstructing the SA from
      the "ticket"

   This memo follows the below guidelines:

o event driven protocol -- no polling involved
o re-create SA's instead of deleting them upon error
o let the side that still has the SA's negotiate fresh SA's after a
   failure
o do not generate state when it can be avoided; reduce CPU cost

2.5. Protocol design rationale

   IKEv2 already specifies a poll-based peer liveness detection
   mechanism. While this type of mechanism helps recovery in most
   situations, the time taken for recovery tends to be high.
   Convergence time requirements are getting shorter and faster
   protocols are becoming a necessity.

   The protocol in this memo is triggered when dangling SA's are
   detected, i.e. when a peer receives unrecognized SPI's. This event
   is in turn triggered when there is actual traffic to be sent so there
   would be little point in just deleting SA's then hoping for the
   systems to recreate them. Instead, these SA's SHOULD be repaired as
   fast as possible in order for the underlying network traffic to be
   forwarded. This protocol assumes that the dangling SA's are meant to
   be rebuilt and not deleted.

   The device that has the SA's also has all the information needed to
   rekey them and becomes the defacto initiator at the end of the
   recovery procedure. This is particularly important for systems with
   dynamic security policies that do not specify how to build the SA; it
   may not be obvious for those peers to determine which security
   parameter they should use to recreate the SA they are now missing.
   When recreating the SA, the peer that has SA's implicitly knows what
   to rebuild and can use the old SA as a template.

   The choice of the rekeyer also brings in an added security value.
   The side that wants to transmit data or at least that pretends having
   SA's has to demonstrate 'willingness' to actually transmit.
   Correspondingly it also means that the gateway that does not have
   SA's is not forced to negotiate anything it may not need. It is
   important to note that the initial effort of setting up timers and
   retransmitting, etc... is left to the side that wants to transmit
   data.

   Last but not least, the protocol can remain stateless until
   sufficient proof of liveness is discovered. In fact, one of the
   protocol variations in this memo allows full statelessness at the
   expense of a round trip time. In an other variation, some small but
   reboot-resistant storage (a key) is used to accelerate the recovery.

3. IKE recovery

3.1. IKE Recovery options

During their IKEv2 exchange, two peers negotiate support for IKE
Recovery. If both peers can store ephemeral information as well as
longer term additional information related to IKE Recovery, an
accelerated procedure for setting up new SAs can be used. This
procedure is called Ticket Based IKE Recovery and is described in
Section 3.3.

If either peer cannot store ephemeral or long term information, peers
fall back to Sateless IKE Recovery described in Section 3.2.

In either case, IKE Recovery is negotiated during the initial IKE
exchange by advertising capabilities as described in Section 5.

3.2. Stateless IKE Recovery

3.2.1. Introducing CHECK_SPI

In order to achieve stateless IKE recovery, this memo introduces a
new notify type called CHECK_SPI. The CHECK_SPI payload carries an
SPI (IKE_SA or Child SA) and one of three sub-types (QUERY, ACK,
NACK). The semantic of the CHECK_SPI subtypes is the following:
o QUERY: a peer queries the remote peer SA DB for the presence of
   the SA whose value is in the payload
o ACK: a peer confirms it has the SA specified in the payload
o NACK: a peer confirms it does not have the SA specified in the
   payload

The payload format of the CHECK_SPI notify is covered in Section 6.

3.2.2. Stateless recovery by invalid IKE packets

When an IKE peer X receives an IKE packet with an unknown IKE SPI
(A,B), that is not an initialization offer (IKE_SA_INIT), peer X
SHOULD send an unprotected INVALID_IKE_SPI notification.

```
Peer X                                                    Peer Y

         HDR(A,B) ...
        <-------------------------------------------

         HDR(A,B) INVALID_IKE_SPI(A,B)
        ------------------------------------------->
```

Even if another IKE_SA exists with the remote peer Y, the

notification MUST NOT be sent protected since peer Y may not share
this SA either.

In order to limit the risk of Denial of Service attacks, the sending
of the INVALID_IKE_SPI notification MUST be rate limited.

When peer Y receives the unauthenticated INVALID_IKE_SPI referencing
the offending IKE SPI (A,B), Y MUST perform the following actions:
o verify that (A,B) is indeed an active IKE_SPI with X
o send to X a notify type CHECK_SPI(QUERY, (A,B), Cookie_Y)

Peer X   Peer Y

         HDR(A,B) INVALID_IKE_SPI(A,B)
        --->

         HDR(A,B) CHECK_SPI(QUERY(A,B),Cookie_Y)
        <---

The sending of the CHECK_SPI packet MUST be rate limited on a per
peer basis.

State SHOULD NOT have been generated by either X or Y at this point.
If the INVALID_IKE_SPI or CHECK_SPI notification gets lost, and X
indeed does not have the IKE SPI, the process will start over again
at the next protected IKE message sent by Y to X.

When peer X receives an unauthenticated CHECK_SPI(QUERY,(A,B))
packet, it MUST perform a look up for (A,B) in its IKE_SA database.
Depending on whether X has or does not have the offending SA, it
SHOULD reply with an IKE packet CHECK_SPI(ACK/NACK,(A,B)). The
cookie data in the CHECK_SPI(ACK/NACK) packet is the same as that
recieved in the CHECK_SPI(QUERY), i.e. the cookie is reflected back
in the response.

Section 3.2.4 discusses cookie generation in greater detail. For
now, it is enough to know that the cookie should contain enough
information for peer Y to validate the CHECK_SPI(ACK/NACK) response
without having to keep any state.

Peer X   Peer Y

         HDR(A,B) CHECK_SPI(QUERY,(A,B),Cookie_Y)
        <---

         HDR(A,B) CHECK_SPI(ACK/NACK,(A,B),Cookie_Y)
         [N(Cookie_X)]
        --->

When peer Y receives the CHECK_SPI(ACK/NACK, Cookie_Y) packet, it
MUST ensure Cookie_Y is valid. If it is not, the packet MUST be
dropped and a rate limited message MUST be logged.

If Cookie_Y is valid and the remote peer X confirms it has the IKE
SPI (via CHECK_SPI(ACK,...)), a rate limited message SHOULD be
logged; this could be a race condition or an attack from a spoofing
attacker.

If Cookie_Y is valid and the remote peer X confirms it does NOT have
the IKE SPI (via CHECK_SPI(NACK,..)), peer Y SHOULD initiate a new
IKE exchange to renegotiate the Parent SA. The parameters of the
negotiation SHOULD be taken primarily from the configuration
(security policy) and, if absent, taken from the confirmed dangling
SA. Renegotiation of CHILD_SA's SHOULD follow the Parent IKE_SA
creation. The original SA's SHOULD be deleted after successful
creation of the new SA's.

Peer X can also include an optional N(Cookie_X) payload in the
CHECK_SPI(ACK/NACK) packet. This Cookie MUST be reflected back by Y
in the new IKE exchange that completes the recovery. This additional
cookie saves one round trip between peers that require an anti-
spoofing Cookie exchange.

A complete recovery exchange for IKE SA's would look like:

Peer X   Peer Y

         HDR(A,B) ...
        <---

         HDR(A,B) INVALID_IKE_SPI(A,B)
        --->

         HDR(A,B) CHECK_SPI(QUERY,(A,B), Cookie_Y)
        <---

         HDR(A,B) CHECK_SPI(NACK,(A,B), Cookie_Y)
         [N(Cookie_X)]
        --->

         HDR(A',0) SAi1, KEi, Ni, [N(Cookie_X)]
        <---

                          ...

3.2.3. Wait before rekey

   There exists a particular attack where a man-in-the-middle can snoop
   and inject traffic but can not block or drop packets. This attack
   can spoof INVALID_SPI (allegedly from X), forcing a CHECK_SPI(QUERY)
   from Y. The attacker would spoof back CHECK_SPI(NACK) to force an
   undue rekey. Since the attacker can not block packets, the
   CHECK_SPI(QUERY) will also reach X, who will reply with
   CHECK_SPI(ACK).

   Y receives CHECK_SPI(NACK) first and MAY wait for a few msec before
   creating a new SA. Y will eventually receive BOTH a CHECK_SPI(ACK)
   and a CHECK_SPI(NACK). Which is dubious. The SIR process should
   then stop and log an error, saving the SA.

   The process is illustrated below:

```
      X                      Attacker                   Y
                             Inv SPI
                             ------------------>

                                CHECK_SPI(QUERY)
          <------------------------------------

                             CHECK_SPI(NACK)
                             ------------------> Should rekey
                                                 but wait a few msec

          CHECK_SPI(ACK)
          ------------------------------------> Hint of attack
                                                 => no rekey
```
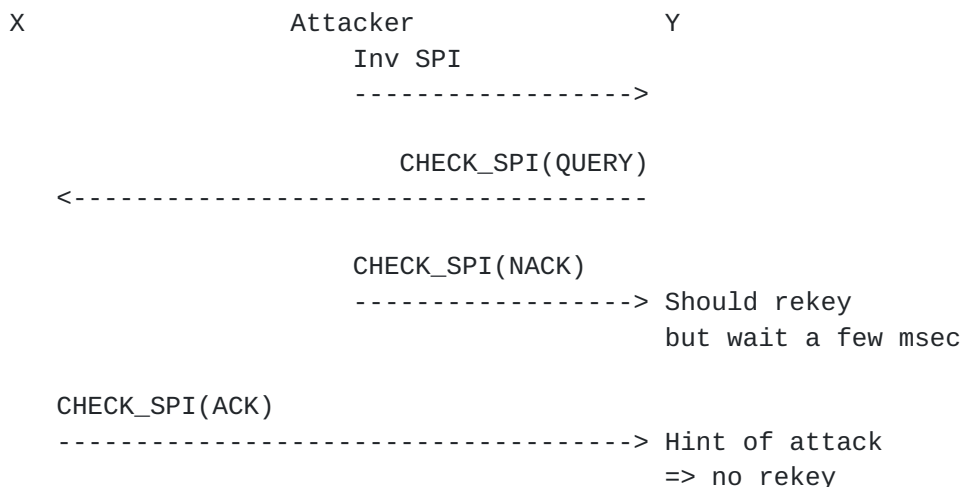
   Ideally, the round-trip-time should be measured during the IKE
   exchange and Y wait for a full RTT before initiating a rekey.

   Given that IKE itself is subject to DH computation by a man-in-the-
   middle, also considering that SA's are dampened after creation (see
   Section 4.2), the staging complexity and limited interest of this
   attack makes it rather impractical. An implementation MAY decide to
   implement this final safety delay but this is strictly optional.

3.2.4. Stateless IKE Recovery cookie

   The cookie information is chosen by the peer that emits it. As such,
   the cookie has strictly no meaning for the remote peer and can thus
   be chosen as seen fit. This section provides recommendations on how
   to generate and validate those cookies.

When an IKE endpoint sends an unauthenticated CHECK_SPI, the cookie
payload following the notify is computed as follow:

                Cookie = <VersionIDofSecret>
                         | H(<secret> | CHECK_SPI(..., Query)
                         | ip.src | ip.dst
                         | udp.src | udp.dst)

where
o <secret> is a randomly generated secret known only to the
   responder and periodically changed
o <VersionIDofSecret> should be changed whenever <secret> is
   regenerated
o CHECK_SPI(..., Query) is the content of the CHECK_SPI notify
   payload where the operation subtype has been set to Query (cf.
   Section 6)
o ip.src is the source ip address of the IKE packet
o ip.dst is the destination ip address of the IKE packet
o udp.src is the source udp post of the IKE packet
o udp.dst is the destination udp port of the IKE packet

Upon reception of a CHECK_SPI(ACK or NACK) response containing a
cookie, a peer can verify whether this is the reply to a Query it
placed by recomputing the cookie and comparing it to the cookie in
the CHECK_SPI message.

In order to minimize the range of cryptographic attacks on <secret>,
messages SHOULD have a limited life time.

3.3. Ticket based IKE recovery using Session Resumption

3.3.1. Ticket Based Recovery

If both peers can store ephemeral information and support IKE Session
Resumption as described in [IKERESUME], an accelerated procedure can
be used. This procedure is called Ticket Based IKE Recovery.

The ticket based IKE Recovery method relies on an unauthenticated
INVALID_IKE_SPI along with a cookie for detection of a dangling SA.
Recovery is effected using session resumption exchange described in
[IKERESUME]to recover from a Dangling SA condition. This memo
introduces a variation to the Session Resumption Exchange for
protection against Denial of Service Attacks

3.3.2. Choice of Recovery Mechanism

The choice of using Stateless IKE Recovery or Ticket Based Recovery
depends upon the capabilities of the endpoint and its peer as well.

It could also depend on policy.

During Recovery, the endpoint that has the SA, also knows about the peers capabilities whereas the enpoint that has lost its SA can be presumed to not know its peers capabilities. This endpoint only offers a hint of its capabilities by responding to an invalid packet with an INVALID_SPI followed by a cookie.

The endpoint that has the SA can choose to respond to an unauthenticated INVALID_SPI based on its knowledge of the peer capabiliries. If it has a session resumption ticket from the peer, it SHOULD initiate an IKE_SESSION_RESUME exchange, else it SHOULD send a CHECK_SPI query. If the peer is not capable of Safe IKE Recovery, the endpoint SHOULD fall back to liveness checks or other mechanisms recommended by [IKEv2].

If the endpoint that receives an IKE_SESSION_RESUME packet is unable to use the resumption ticket for any reason, it should respond with a RESUME_NACK followed by the peer coookie it recieved in the clear. This allows the peer to initiate a full IKEv2 exchange safely.

3.3.3. Ticket based recovery by invalid IKE packets

When a peer X receives an IKE packet with an unknown IKE_SPI, it SHOULD send an unprotected INVALID_IKE_SPI notify to the sender Y. The INVALID_IKE_SPI MUST be followed with a Cookie payload. The cookie payload content is relevant only to the generator of the cookie and a suggested format for it is described in Section 3.2.4.

When peer Y receives the INVALID_IKE_SPI referencing the IKE_SPI(A,B) followed by CHECK_SPI(Cookie_X), it MUST perform the following actions:
o verify that (A,B) is an active IKE_SA it has with X. If no such SA exists a rate limited mesage SHOULD be logged.
o verify that it possesses a resumption ticket given to it by X and initiate an IKE_SESSION_RESUME exchange with X. This memo requires that the IKE_SESSION_RESUME packet MUST carry COOKIE_X received in the INVALID_SPI packet encrypted in the SK payload. Y also generates and sends another cookie COOKIE_Y in the clear.

```
   Peer X                                                       Peer Y

                                               HDR(A,B) ...
         <----------------------------------------------


   (1)   HDR(A,B) INVALID_IKE_SPI(A,B)
         CHECK_SPI(COOKIE_X)
         ---------------------------------------------->

   (2)               HDR(A,B) Ni CHECK_SPI(COOKIE_Y)
          N(TICKET) SK{IDi, IDr... CHECK_SPI(COOKIE_X)}
         <----------------------------------------------

   (3)   HDR(A,B) SK{Nr,IDr,SAr2... CHECK_SPI(COOKIE_Y)}
         ---------------------------------------------->

                                               SK{}
   (4)   <----------------------------------------------
                           ...
```

At step(1), If Peer Y does not support [IKERESUME], it MUST ignore
CHECK_SPI(COOKIE_X) and fall back to the stateless recovery method in
Section 3.2. Otherwise, on step (2), Peer Y responds to the invalid
SPI by sending back the resumption ticket as well as COOKIE_X under
the cover of SK.

Peer X, on receiving the TICKET_RESUME notify with a cookie payload
on step (2) MUST look up the SA (A,B) in its SA database. If the SA
exists, it MUST respond with a protected CHECK_SPI(ACK) that includes
the peer cookie COOKIE_Y and a rate limited message SHOULD be logged.

If the SA does not exist, X should decrypt the SK payload using the
contents of the ticket and validate COOKIE_X. If the cookie is not
valid the packet should be dropped and a rate limited message SHOULD
be logged.

If packet (2) is rejected for any other reason, Peer X responds with
a CHECK_SPI(NACK) containing COOKIE_Y and the exchange continues
statelessly as described in Section 3.2.

If packet (2) is finally accepted and validated, peer X sends back an
IKE_SESSION_RESUME response to create a new SA. The response packet
also includes CHECK_SPI(COOKIE_Y) which is simply sent back unchanged
but protected inside the SK payload. Peer X can also proceed to
computing and creating state for a new SA as described in
[IKERESUME]. A further cookie exchange as described in [IKERESUME]
is not required as the two peers have already reflected COOKIE_X.

   Peer Y performs the following actions on Packet (3) depending on the
   response it gets back from X
   o On receiving a SESSION_RESUME response, Peer Y decrypts the SK
      payload and validates the COOKIE2, and proceeds to create a new
      SA. If the cookie is invalid a rate limiting message is logged
      and the packet is dropped.
   o If the Peer Y receives a CHECK_SPI(NACK) followed by the cookie
      COOKIE_Y, Y MAY proceed to initiating a regular IKEv2 session.
   o If a protected CHECK_SPI(ACK) response is received, a rate
      limiting message is logged.
   o If the Peer Y receives a N(TICKET_NACK) notification, Y MAY
      initiate a regular IKEv2 exchange.

   Packet (4) is an empty informational liveliness message sent from Y
   to X using the newly installed SA, after a successful SESSION_RESUME
   exchange.

   Note: It is to be noted that the Stateful Ticket Recovery exchange
   described above could be used without any change, to complete a
   recovery even when if the QCD_TOKEN based method described in [QCD]
   is used for detection of Invalid SAS.

3.4. IPsec SA recovery

   We are now considering the case of an IKE endpoint Y sending an ESP
   or AH packet (or any type of traffic supported by a CHILD_SA) to peer
   X who does not have the corresponding phase 2 SA. We will
   differentiate two subcases depending on the presence or not of an IKE
   SA between the two peers.

   The recovery procedure will be roughly the same as for the Dangling
   Parent SA case but for children SA's, we send protected notifications
   whenever we can.

   Peer X   Peer Y

            ESP(SPI) ...
          <--

   On receiving an unrecognized ESP or AH packet, Peer X SHOULD notify
   the remote peer Y. The method will be different, according to the
   presence of an IKE_SA with Y.

3.4.1. In the presence of an IKE_SA

   In IKEv2, when an IKE_SA is available between two peers, CHILD_SA's
   SHOULD not be out of sync thanks to the acknowledgement and
   retransmissons of notifies. IKEv2 however does not specify what to

do when a peer does not eventually respond to protected DELETE_SPI
notifies.

This section augments the IKEv2 specification in order to allow the
recovery of stale SA's in case peers decided to keep the Parent SA
nevertheless.

If an IKE_SA is available with the remote peer, peer X MUST send a
protected INVALID_SPI notification to the Y. The notification MUST be
protected by the Parent SA and MUST contain the SPI of the invalid
packet.

```
Peer X                                                   Peer Y

         ESP(SPI) ...
        <-------------------------------------------

         HDR(A,B) SK{INVALID_SPI(SPI)}
        ------------------------------------------->
```

At this point, Y MUST check whether it has the offending SA. If so,
it SHOULD re-key or delete the child SA according to its security
policy. This document suggests that Y SHOULD delete the dangling SA
but MAY rekey if deemed adequate. If the offending SA is not to be
found, a message SHOULD be logged as the triggering ESP packet may be
an attack or the result of a race condition. The logging MUST be
rate limited.

3.4.2. In the absence of an IKE_SA

If an IKE_SA is not available with peer Y, an unprotected INVALID_SPI
notification MUST be sent. The notification MUST contain the SPI of
the invalid packet.

```
Peer X                                                   Peer Y

         ESP(SPI) ...
        <-------------------------------------------

         HDR(0,0) INVALID_SPI(SPI)
        ------------------------------------------->
```

Note: An IKE SPI of (0,0) is used since there is no other IKE SPI to
use (by construction)

Peer Y MUST verify whether it has the offending CHILD_SA; if it does
not, Y MUST log a rate limited message and drop the notify. If Y
owns the offending SA, Y MUST perform the following:

o   ensure the unauthenticated INVALID_SPI notify is legitimate
o   rebuild the dangling SA's with the remote peer if needed
The following procedure will help determining whether the INVALID_SPI
notify is legitimate.

Peer Y MUST send a protected CHECK_SPI notify to X. Since Y has the
CHILD_SA, it MUST have its Parent SA by construction.

Peer X   Peer Y

        HDR(0,0) INVALID_SPI(SPI)
        -->

        HDR(A,B) SK{CHECK_SPI(QUERY, SPI)}
       <--

If X can decrypt the CHECK_SPI(QUERY) notification from Y, i.e it has
a valid IKE_SA(A,B), the situation can be either of the following:
o   therY also generates and sends another cookie COOKIE_Y in the
    cleare is a logic error on X as it should have sent the
    INVALID_SPI protected
o   the INVALID_SPI request that led to the CHECK_SPI notify has been
    forged
o   there was a race condition in an earlier exchange

X MUST try to identify which condition it has met, e.g. by checking
SPI is in the SA database and MUST log a message about a possible
security alert.

Under normal recovery circumstances, X will not have the PARENT SA.
In this case, X MUST reply with an unprotected INVALID_IKE_SPI(A,B)
and fall back into the Parent SA recovery procedure.

The Parent SA recovery procedure could use either stateless or Ticket
based recovery. The overall recovery scheme for CHILD_SA's using the
Stateless IKE recovery procedure can be summarized as follows:

```
Peer X                                                        Peer Y

            ESP(SPI) ...
          <--------------------------------------------

           HDR(0,0) INVALID_SPI(SPI)
          -------------------------------------------->

           HDR(A,B) SK{CHECK_SPI(QUERY,(SPI)) }
          <--------------------------------------------

           HDR(A,B) INVALID_IKE_SPI (A,B)
          -------------------------------------------->

           HDR(A,B) CHECK_SPI(QUERY,(A,B),Cookie)
          <--------------------------------------------

           HDR(A,B) CHECK_SPI(NACK,(A,B),Cookie)
          -------------------------------------------->

           HDR(A',0) SAi1, KEi, Ni
          <--------------------------------------------
```

3.5. Mandatory Initiators

There are cases where the side having the SA's cannot act as an
initiator in a recovery procedure and has to rely on the peer device
to initiate recovery. These exceptions include:
o Specific implementations, typically in remote access, that rely on
   the 'client' to be a pure initiator.
o gateways that are behind a dynamic PAT device and that can not be
   reached directly from outside. These devices have to be
   initiators of the connection in order to set up the translation
   rules.

We call such devices Mandatory Initiators and in the context of this
document, they will eventually become responsible for recovering the
SA's.

Mandatory Initiators SHOULD be determined by the system administrator
through their configuration or implicitly through the set of features
they are configured for. Mandatory Initiators MAY determine by
themselves whether they are behind a dynamic PAT device. The
determination can for instance arise from analyzing the NAT-T
payloads described in [NAT-T].

Because Mandatory Initiators are actually IKEv2 initiators, they
typically know by configuration which peers they should have a

connection with, even if the SA's are missing. If this is indeed the
case, the following Mandatory Initiator recovery procedure SHOULD be
followed.

The recovery procedure for Mandatory Initiators is the same as for
other peers with change in the last step containing the
CHECK_SPI(NACK) where the Mandatory Initiator actually sends
initiates an an IKEv2 Initial Exchange along with the CHECK_SPI(NACK)
payload.

Example CHILD_SA recovery exchange with mandatory initiator (Parent
SA present):

```
Peer X                                                  Peer Y

                                    HDR(A,B) ...
          <--------------------------------------------

          HDR(A,B) INVALID_IKE_SPI(A,B)
          -------------------------------------------->

           HDR(A,B) CHECK_SPI(QUERY,(A,B), Cookie_X)
          <--------------------------------------------

          HDR(A',0) SAi1, KEi, Ni,
          CHECK_SPI(NACK,(A,B), Cookie_X)
          -------------------------------------------->

            ...
```

When Peer Y receives the Initial Offer, it MUST verify it has the IKE
SPI in the CHECK_SPI reply. In other words, the recovery procedure
HINTS the Mandatory Initiator about a need for resynchronizing the
SA's. This hint MAY be ignored, according to the local peer policy.

If it does not have the corresponding IKE SA, Y MUST log a rate
limited message and drop the message. If Y owns the IKE SPI, it MUST
validates the cookie as described in Section 3.2.4 and proceed with
the IKE exchange, according to its security policy.

In any case, X SHOULD NOT retransmit the Initial Offer. The process
will restart by itself if the IKE SA is indeed missing and further
offending ESP or IKE packets are emitted. If X receives a valid
Message 2, it can proceed with the rest of the IKEv2 negotiation and
retransmit as necessary.

Example CHILD_SA recovery exchange with mandatory initiator (no
Parent SA):

```
   Peer X                                                  Peer Y
   (Mandatory Initiator)

           ESP(SPI) ...
         <--------------------------------------------

          HDR(0,0) INVALID_SPI(SPI)
         --------------------------------------------->

          HDR(A,B) CHECK_SPI(QUERY,(SPI))
         <--------------------------------------------

          HDR(A,B) INVALID_IKE_SPI (A,B)
         --------------------------------------------->

          HDR(A,B) CHECK_SPI(QUERY,(A,B), Cookie)
         <--------------------------------------------

          HDR(A',0) SAi1, KEi, Ni,
          CHECK_SPI(NACK,(A,B),Cookie)
         --------------------------------------------->
```

[3.6](#). Recovery closure

   In many cases, the outcome of the recovery procedure yields to the
   creation of a new IKE_SA. Either side may be left with an old IKE_SA
   and dangling CHILD_SA's. In order to recover entirely, the old
   CHILD_SA's SHOULD be recreated (entirely renegotiated) under the
   protection of the new Parent SA. After which, the old SA's (IKE_SA
   and CHILD_SA's) SHOULD be entirely deleted.

[3.7](#). Dealing with race conditions

   When a peer deletes SA's, a DELETE payload is sent that MUST be
   acknowldeged. Before the delete notify reaches the remote peer,
   further ESP packets for the now deleted SPI may be received. These
   ESP packets MUST be silently discarded as long the DELETE Notify can
   be retransmitted.

[4](#). Throttling and dampening

   An important aspect of the security in IKE recovery has to do with
   limiting the CPU utilization. In order to thwart flood types
   denial of service attacks, strict rate limiting and throttling
   mechanisms have to be enforced.

   All the notifications that are exchanged during IKE recovery SHOULD

be rate limited. This paragraph provides information on the way rate
limiting should take place.

4.1. Invalid SPI throttling

The sending of all Invalid SPI notifies MUST be rate limited one way
or an other. The rate limiting SHOULD be performed on a per peer
basis but dynamic state creation SHOULD be avoided as much as
possible (it can be; this is an implementation matter).

Invalid SPI rate limiting protects against natural dangling SA
occurences. I.e. normal traffic conditions may cause unrecognized
SPI's to be received and this message is the most important to
protect. Indeed, it is not realistic to send one notification per
bad ESP packet received. On high speed links, this could mean
thousands of IKE notifies sent for the same offending SPI.

The receiving of unauthenticated Invalid SPI notifies MUST as well be
rate limited. Again, the rate limiting SHOULD be performed on a per
peer basis without dynamic state creation. In normal circumstances,
the peer receiving Invalid SPI notifies has an SA with the peer
sendig those notifies and already maintains peer-related data
structures that can help in maintaining adequate counters.

Authenticated Invalid SPI notifies can be accepted without
throttling.

4.2. Dampening

After one of the following conditions:
o the natural creation or rekey of one or more SA's
o the recovery of one or more SA's
o the failure in recovering an SA owned by the local security
   gateway
o the logging of an error or warning message involving an SA owned
   by the local security gateway

The peer with which SA's were created, attempted or against which a
log was emitted SHOULD be dampened, which means that all the
unauthenticated Invalid SPI and Check SPI messages emitted by that
peer MUST be ignored for a chosen duration.

This protection prevents a man-in-the-middle from forcing the fast
recreation of SA's and potentially depleting the entropy of systems
under attack. It also deals efficently with race conditions that may
occur after a rekey.

4.3. User controls

   Because throttling at large is related to speed, the network
   implementation around the security gateways has a major influence on
   the pertinence of the paremeters controlling rate limiting. It is
   difficult to provide good absolute values for the rate limiters,
   considering that these are implementation dependent.

   As such, for the sake of fitness in practical deployments, a system
   implementing this memo MUST provide administrative controls over the
   rate limiter parameters.

5. Negotiating IKE recovery

   IKE recovery capabilities MUST be advertised through a Vendor ID
   payload.

   In the first two messages of the Parent SA negotiation, the Vendor ID
   payload for this specification MUST be sent if supported (and it MUST
   be received by both sides). The content of the payload is the ASCII
   string:

   SECURE IKE RECOVERY, or in hex: 53 45 43 55 52 45 20 49 4b 45 20 52
   45 43 4F 56 45 52 59"

   The peers' capbility for IKE Session Resumption is known implicitly
   from receiving the resumption ticket.

   Determining peer capability can be useful for two reasons at least.
   First, this information MAY let a system decide to fallback to
   another recovery mechanism, such as from Ticket based Recovery to
   Stateless Safe IKE Recovery or falling back to the one embedded in
   IKEv2

   Knowledge of the peer's capabilities can be used by the 'live
   peer'(the one that still has the SA's) in order to determine whether
   it is normal or not to receive unauthenticated INVALID_SPI with or
   without cookies or CHECK_SPI notifies. A peer that has lost
   information about it's peer SHOULD go under the assumption that peer
   does understand IKE Recovery as described in this memo. This
   assumption implies that INVALI_SPI notifies with cookies and
   CHECK_SPI notifies can be sent. If the remote peer does not support
   IKE Recovery, it will just ignore these messages.

   In general, it is useful for system administrators to monitor the
   capabilities of a remote system connecting to a local security
   gateway and there is an interest in advertising the IKE Recovery

   capability.

6. Payload formats

   For reference, the Notify Payload is defined as follow

```
    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  !   SPI Size    !      Notify Message Type      !
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   !                                                               !
   ~               Security Parameter Index (SPI)                 ~
   !                                                               !
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   !                                                               !
   ~                      Notification Data                        ~
   !                                                               !
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

   The meaning of the fields is the same as defined in [IKEv2].

   This memo introduces a new Notify Message Type that is being
   developped with a Private Use Type:
   o   CHECK_SPI: 32770

   An official IANA assigned number MUST be assigned if this document
   reaches final recommendation state.

   The notification data area is formatted as such:

```
                        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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   ! Subtype       ! CookieSize    |            RESERVED           !
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   !                            Cookie                             !
   ~                                                               ~
   !                                                               !
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

   o   Subtype (1 Octet) - This field determines the operation being
       performed (Query, Reply_ACK, Reply_NACK, Lone_Cookie)
   o   CookieSize - The size of the optional cookie embedded in the
       CHECK_SPI notify. If there is no cookie, CookieSize is set to 0
       (zero).

   o Cookie - The optional cookie embedded in the payload. Its size
      depends on the value CookieSize.

   The list of operations and their corresponding value:
   o Query: 0
   o Reply_ACK: 1
   o NACK: 2

7. IANA Considerations

   This document requires the following notification to be registered by
   IANA. The corresponding registry was established by IANA.
   o CHECK_SPI Notification type (Section 6).

8. Security Considerations

   IKE recovery self-protection is discussed all along the document and
   contains many mechanism to thwart denial of service attacks.

   IKE recovery is subject to a man-in-the-middle attack that can let
   the attacker trigger a renegotiation. It has to be noticed that an
   attacker able to block ESP and/or IKE packets can cause IKE itself to
   also tear down and trigger a rekey of IKE SA's. With throttling and
   dampening enabled, IKE recovery is able to reduce the amount of
   rekeys/negotiations to as low a rate as IKEv2.

   Overall, IKE Recovery is not more vulnerable than IKEv2 and even
   improves on the security of IKEv2 by resynchronizing SA's more
   rapidly which is important with dynamic polices.

9. Collapsed stateless exchange

   In order to save on round-trips, IKE Recovery can be collapsed into
   an IKEv2 exchange. The recovery case goes as follows:

```
   Peer X                                              Peer Y

   (1)                                      HDR(A,B) ...
                <----------------------------------------------

   (2)        HDR(A,B) INVALID_IKE_SPI(A,B), N(Cookie_X)
                ---------------------------------------------->

   (3)                            HDR(A',0) HDR N(Cookie_X),
            CHECK_SPI(QUERY,(A,B)), N(Cookie_Y), SAi1, KEi, Ni
                <----------------------------------------------

   (4)        HDR(A',B') CHECK_SPI(NACK,(A,B)), N(Cookie_Y),
              SAr1, KEr, Nr, [CERTREQ], KEi, Ni
                ---------------------------------------------->

   (5)        HDR(A',B') SK {IDi, [CERT,] [CERTREQ,] [IDr,]
                                        AUTH, SAi2, TSi, TSr}
                <----------------------------------------------

                            ...
```

   In the exchange above, X sends an optional N(Cookie_X) in the
   INVALID_IKE_SPI notify if it expects cookies to be used when acting
   as a responder. Cookie_X is reflected by peer Y as in a normal IKE

10. Change log

   NOTE TO RFC EDITOR: This section is to be removed in the final RFC

10.1. Changes from draft-fdetienn-sir-02

   Minor changes around rate limiting. Removed implementation
   recommendation to keep only high level recommendation.

11. References

11.1. Normative References

   [Bra97]   Bradner, S., "RFC 2119, Key Words for use in RFCs to
              indicate Requirement Levels", March 1997.

   [IKEv2]   Kaufman, Ed., "RFC 4306, Internet Key Exchange (IKEv2)
              Protocol", December 2005.

   [NAT-T]   Kivinen, "RFC 3947, Negotiation of NAT-Traversal in the

IKE", January 2005.

11.2. Informative References

[BIRTHCERT]
           Harkins, D., Kauffman, C., Kivinen, T., Kent, S., and R.
           Perlman, "Design Rationale for IKEv2", July 2007.

[IKERESUME]
           Sheffer, Y., "Stateless Session Resumption for the IKE
           Protocol", July 2007.

[QCD]   Nir, Y., "A Quick Crash Detection Method for IKE",
           Aug 2008.

Authors' Addresses

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

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

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

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

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

   Email: yir@checkpoint.com