Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 12, 2011

Y. Nir
Check Point
F. Detienne
P. Sethi
Cisco

July 11, 2010

A Quick Crash Detection Method for IKE
draft-nir-ike-qcd-07

Abstract

This document describes an extension to the IKEv2 protocol that allows for faster detection of 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

Copyright Notice

Table of Contents

1. Introduction

IKEv2, as described in [RFC4306] 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, or else the non-rebooted peer has a dynamic IP address. In such cases, the rebooted peer will not be able to re-establish the tunnels. Section 2 describes how recovery works under RFC 4306, 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 is 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 9.2.

2. RFC 4306 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 [RFC4306]. 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)."
        XML2PDFRFC-ENDARTWORK
```

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 [RFC4306]. 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..."
        XML2PDFRFC-ENDARTWORK
```

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

3. Protocol Outline

Supporting implementations will send a notification, called a "QCD token", as described in Section 4.1 in the last IKE_AUTH exchange messages. These are the final 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) as part of the IKE SA. Implementations that support this part of the protocol will be called "token takers". Section 9.1 has considerations for which implementations need to be token takers, and which should be token makers. Implementation 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 MUST 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 in 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 dependant 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.

There is ongoing work on IKEv2 Session Resumption ([resumption]). See Section 8 for a short discussion about this extensions´s interaction with session resumption.

4. Formats and Exchanges

4.1. Notification Format

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

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Next Payload  !C!  RESERVED   !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!  Protocol ID  !   SPI Size    ! QCD Token Notify Message Type !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                                                               !
~                       TOKEN_SECRET_DATA                       ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      XML2PDFRFC-ENDARTWORK
```

- 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 [RFC4306].
- 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 (16-128 octets) 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 A.3 of [RFC4718].

```
 first request        --> IDi,
                         [N(INITIAL_CONTACT)],
                         [[N(HTTP_CERT_LOOKUP_SUPPORTED)], CERTREQ+],
                         [IDr],
                         [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
                         [N(QCD_TOKEN)]

 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+]
          XML2PDFRFC-ENDARTWORK
```

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 MUST 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)
          XML2PDFRFC-ENDARTWORK
```

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

```
   request             --> N(QCD_TOKEN)

   response            <--
          XML2PDFRFC-ENDARTWORK
```

For session resumption, as specified in [resumption], 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.

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)] }  -->
          XML2PDFRFC-ENDARTWORK
```

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 INFORMATIONAL Exchange

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.

```
         request             --> N(INVALID_IKE_SPI), N(QCD_TOKEN)+
          XML2PDFRFC-ENDARTWORK
```

If child SPIs are persistently mapped to IKE SPIs as described in Section 9.2, a token taker may get the following unprotected message in response to an ESP or AH packet.

```
         request             --> N(INVALID_SPI), N(QCD_TOKEN)+
          XML2PDFRFC-ENDARTWORK
```

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 [RFC4306], 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, an empty response MUST be sent. If the QCD_TOKEN cannot be validated, a response MUST NOT be sent. Section 5 defines token verification.

5. Token Generation and Verification

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

The following lists the requirements from 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.

5.1. A Stateless Method of Token Generation

This describes a stateless method of generating a token:

- 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)

 XML2PDFRFC-ENDARTWORK
```

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.

The reason to use this method is described in Section 9.3. 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)

          XML2PDFRFC-ENDARTWORK
```

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

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.

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. This is also different from high availability configurations where the SAs are synchronized. For load sharing configurations, please see Section 10.2 for security considerations.

7. Alternative Solutions

7.1. 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.

7.2. 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.
          XML2PDFRFC-ENDARTWORK
```

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.

7.3. Reducing Liveness Check Length

Some have suggested that the RFC 4306 procedure described in Section 2 can be tweaked by requiring 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, reducing elapsed time and retransmission count will 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 until the tunnels are active, or from the time the backup gateway has taken over until the tunnels are active.

8. Interaction with Session Resumption

Session Resumption, specified in [resumption] proposes to make setting up a new IKE SA 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 would require all these many remote access clients to establish an IKE SA either with the rebooted gateway or with a backup gateway. This tunnel re- establishment should occur 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 as long as the lifetime of a child SA, because IPsec does not have packet acknowledgement, and applications cannot signal the IPsec layer that the tunnel "does not work". Before establishing a new IKE SA using Session Resumption, a client should ascertain that the gateway has indeed failed. This could be done using either a liveness check (as in RFC 4306) or using the QCD tokens described in this document.

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:

```
     Initiator                 Responder
     -----------               -----------
    HDR, SAi1, KEi, Ni  -->

                        <--    HDR, SAr1, KEr, Nr, [CERTREQ]

    HDR, SK {IDi, [CERT,]
    [CERTREQ,] [IDr,]
    AUTH, N(QCD_TOKEN)
    SAi2, TSi, TSr,
    N(TICKET_REQUEST)}  -->
                        <--    HDR, SK {IDr, [CERT,] AUTH,
                               N(QCD_TOKEN), SAr2, TSi, TSr,
                               N(TICKET_LT_OPAQUE) }

             ---- Reboot -----

    HDR, {}             -->
                        <--  HDR, N(QCD_TOKEN)

    HDR, [N(COOKIE),]
    Ni, N(TICKET_OPAQUE)
    [,N+]               -->
                        <--  HDR, Nr [,N+]

          XML2PDFRFC-ENDARTWORK
```

9. Operational Considerations

9.1. 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 requirements:

- A remote-access client MUST be a token taker and MAY be a token maker.
- A remote-access gateway MAY be a token taker and MUST be a token maker.
- An inter-domain VPN gateway MUST be both token maker and token taker.

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 [RFC4306] and either send an INVALID_IKE_SPI notification, or ignore it entirely.

9.2. 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 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.

9.3. Using Tokens that Depend on IP Addresses

This section describes the rationale for token generation methods such as the one described in Section 5.2. Note that this section merely provides a possible rationale, and does not specify or recommend any kind of configuration.

Some configurations of security gateway use a load-sharing cluster of hosts, all sharing the same IP addresses, where the SAs (IKE and child) are not synchronized between the cluster members. In such a configuration, a single member does not know about all the IKE SAs that are active for the configuration. A load balancer (usually a networking switch) sends IKE and IPsec packets to the several members based on source IP address.

In such a configuration, an attacker can send a forged protected IKE packet with the IKE SPIs of an existing IKE SA, but from a different IP address. This packet will likely be processed by a different cluster member from the one that owns the IKE SA. Since no IKE SA state is stored on this member, it will send a QCD token to the attacker. If the QCD token does not depend on IP address, this token can immediately be used to tell the token taker to tear down the IKE SA using an unprotected QCD_TOKEN notification.

To thwart this possible attack, such configurations should use a method that considers the taker´s IP address, such as the method described in Section 5.2.

10. Security Considerations

10.1. 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, she 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.

10.2. QCD Token Transmission

A token maker MUST NOT send a QCD token in an unprotected message for an existing IKE SA. This implies that a conforming QCD token maker MUST be able to tell whether a particular pair of IKE SPIs represent a valid 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 QCD token for an IKE SA which is valid on another gateway.

This document does not specify how a load sharing sharing configuration of IPsec gateways would work, but in order 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 it is to synchronize a list of active IKE SPIs among all the cluster members.

10.3. 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 is faulty and always generates predictable IKE SPIs, the space is still at least 2^64 entries, so making the dictionary is extremely hard.
- Throttling the amount of QCD_TOKEN notifications sent out, as discussed in Section 9.1, 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.

11. 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".

12. Acknowledgements

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

13. Change Log

This section lists all changes in this document

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

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

Mostly editorial changes and cleaning up.

13.2. 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 dependant on peer IP address and their interaction with MOBIKE.

13.3. 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.

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

- Merged proposal with draft-detienne-ikev2-recovery [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.

13.5. 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.

14. References

14.1. Normative References

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

[RFC4306] Kaufman, C., "Internet Key Exchange (IKEv2) Protocol", RFC 4306, December 2005.

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

[RFC4718] Eronen, P. and P. Hoffman, "IKEv2 Clarifications and Implementation Guidelines", RFC 4718, October 2006.

14.2. Informative References

[recovery] Detienne, F., Sethi, P., and Y. Nir, "Safe IKE Recovery", draft-detienne-ikev2-recovery (work in progress), August 2008.

[resumption]

Sheffer, Y. and H. Tschofenig, "IKEv2 Session Resumption", draft-ietf-ipsecme-ikev2-resumption (work in progress), June 2009.

Authors´ Addresses

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

Email: ynir@checkpoint.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