

Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: September 17, 2008

Y. Nir  
Check Point  
March 16, 2008

A Quick Crash Recovery Method for IKE  
draft-nir-qcr-00.txt

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with [Section 6 of BCP 79](#).

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/lid-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 September 17, 2008.

Copyright Notice

Copyright (C) The IETF Trust (2008).

Abstract

This document describes an extension to the IKEv2 protocol that allows for faster crash recovery using a saved token method.

When an IPsec tunnel between two IKEv2 implementations is disconnected due to a restart of one peer, it can take as much as several minutes to recover. In this text we propose an extension to the protocol, that allows for recovery within a few seconds of the reboot.

Internet-Draft

Quick Crash Recovery

March 2008

## Table of Contents

<a href="#">1.</a>	Introduction . . . . .	<a href="#">3</a>
<a href="#">1.1.</a>	Conventions Used in This Document . . . . .	<a href="#">3</a>
<a href="#">2.</a>	<a href="#">RFC 4306</a> Crash Recovery . . . . .	<a href="#">3</a>
<a href="#">3.</a>	Protocol Outline . . . . .	<a href="#">4</a>
<a href="#">4.</a>	Formats and Exchanges . . . . .	<a href="#">4</a>
<a href="#">4.1.</a>	Notification Format . . . . .	<a href="#">4</a>
<a href="#">4.2.</a>	Authentication Exchange . . . . .	<a href="#">5</a>
<a href="#">4.3.</a>	Informational Exchange . . . . .	<a href="#">7</a>
<a href="#">5.</a>	Token Generation and Verification . . . . .	<a href="#">7</a>
<a href="#">5.1.</a>	A Stateful Method of Token Generation . . . . .	<a href="#">7</a>
<a href="#">5.2.</a>	A Stateless Method of Token Generation . . . . .	<a href="#">8</a>
<a href="#">5.3.</a>	Token Lifetime . . . . .	<a href="#">8</a>
<a href="#">6.</a>	Alternative Solutions . . . . .	<a href="#">8</a>
<a href="#">6.1.</a>	Why not Save the Entire IKE SA . . . . .	<a href="#">8</a>
<a href="#">6.2.</a>	Initiating a new IKE SA . . . . .	<a href="#">9</a>
<a href="#">7.</a>	Operational Considerations . . . . .	<a href="#">9</a>
<a href="#">8.</a>	Security Considerations . . . . .	<a href="#">9</a>
<a href="#">9.</a>	IANA Considerations . . . . .	<a href="#">10</a>
<a href="#">10.</a>	Normative References . . . . .	<a href="#">10</a>
	Author's Address . . . . .	<a href="#">11</a>
	Intellectual Property and Copyright Statements . . . . .	<a href="#">12</a>

## 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 peers have a dynamic IP address. In such cases, the rebooted peer will not re-establish the tunnels.

[Section 2](#) describes the current procedure, and explains why crash recovery can take up to several minutes. The method proposed here, is to send a token in the IKE\_AUTH exchange that establishes the tunnel. That token can be maintained on the peer in some kind of persistent storage such as a disk or a database, and can be used to delete the IKE SA after a crash. Deleting the IKE SA results in a quick re-establishment of the IPsec tunnel.

### 1.1. Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [[RFC2119](#)].

## 2. [RFC 4306](#) Crash Recovery

When one peer reboots, the other peer does not get any notification, so 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 INFORMATIONAL exchange with an 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)."

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 INFORMATIONAL 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". Clearly, implementations differ, but all will take a significant amount of time.

### [3.](#) Protocol Outline

Supporting implementations will send a notification, called a "QCR token", as described in [Section 4.1](#) in the last packets of the IKE\_AUTH exchange. These are the final request and final response that contain the AUTH payloads. The generation of these tokens is a local matter for implementations, but considerations are described in [Section 5](#).

A supporting implementation receiving such a token SHOULD store it in such a way, that it will survive a reboot. When a supporting implementation receives a protected IKE request message with unknown IKE SPIs, it should scan its saved token store. If a token matching the IKE SPIs is found, it SHOULD send it to the requesting peer in an unprotected IKE message as described in [Section 4.3](#).

When a supporting implementation receives the QCR notification token in an unprotected INFORMATIONAL exchange, it MUST verify that the TOKEN\_SECRET\_DATA field is associated with the IKE SPIs in the IKE\_SPI fields of the IKE packet. If the verification fails, it SHOULD log the event. If it succeeds, it MUST delete the IKE SA associated with the IKE\_SPI fields, and all dependant child SAs. This event MAY also be logged.

A supporting implementation 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](#)]. The current proposal is orthogonal to Session Resumption, and in fact using Session Resumption instead of a regular IKE exchange, the new SA can be created with minimal overhead.

## 4. Formats and Exchanges

### 4.1. Notification Format

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

Nir Expires September 17, 2008 [Page 4]

---

Internet-Draft Quick Crash Recovery March 2008

```

                                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    ! QCR Token Notify Message Type !
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
!                                                         !
~                               TOKEN_SECRET_DATA                               ~
!                                                         !
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

- o Protocol ID (1 octet) MUST contain 1, as this message is related to an IKE SA.
- o SPI Size (1 octet) MUST be zero, in conformance with [[RFC4306](#)].
- o QCR Token Notify Message Type (2 octets) - Must be xxxxx, the value assigned for QCR token notifications. TBA by IANA.
- o TOKEN\_SECRET\_DATA (16-256 octets) contains a generated token as described in [Section 5](#).

### 4.2. Authentication Exchange

For clarity, only the EAP version of an AUTH exchange will be presented here. The non-EAP version is very similar. The figure below is 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+]
```

```

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

last request      --> AUTH
                  [N(QCR_TOKEN)]

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

```

Note that the QCR\_TOKEN notification is marked as optional because it is not required by this specification that both sides send QCR tokens. If only one peer sends the QCR 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 QCR\_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 doesn't know whether the other side supports it.

#### [4.3.](#) Informational Exchange

This informational exchange is non-protected, and is sent as a response to a protected IKE request, which uses an IKE SA that is unknown.

```

request          --> N(QCR_TOKEN)

```

response

<--

The QCR\_TOKEN is the only notification in the request. 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.

If the QCR\_TOKEN verifies OK, an empty response MUST be sent. If the QCR\_TOKEN cannot be validated, a response SHOULD NOT be sent. [Section 5](#) defines token verification.

## [5.](#) Token Generation and Verification

No token generation method is mandated by this document. Two methods are documented in [Section 5.1](#) and [Section 5.2](#), but they only serve as examples.

The following lists the requirements from a token generation mechanism:

- o Tokens should be at least 16 octets long, and no more than 256 octets long, to facilitate storage.
- o It should not be possible for an external attacker to guess the QCR token generated by an implementation. Cryptographic mechanisms such as PRNG and hash functions are RECOMMENDED.
- o The peer that generated the QCR token, should be able to immediately verify it, provided that the IKE SPIs are given, and that the IKE SA has not expired or been otherwise deleted.

### [5.1.](#) A Stateful Method of Token Generation

This describes a stateful method of generating a token:

- o Before sending the QCR token, 32 random octets are generated using a secure random number generator or a PRNG.
- o Those 32 bytes are used as the TOKEN\_SECRET\_DATA field, and stored as part of the IKE SA.
- o For verification, the IKE implementation simply retrieves the IKE SA, and compares the TOKEN\_SECRET\_DATA field from the notification to the TOKEN\_SECRET\_DATA field stored with the SA.

### [5.2.](#) A Stateless Method of Token Generation



This describes a stateless method of generating a token.

- o At startup, the IKE implementation generates a 32-octet random buffer using a cryptographically secure PRNG. This buffer is called the QCR\_SECRET.
- o For each QCR token, the TOKEN\_SECRET\_DATA field is generated by calculating a SHA-256 hash over a concatenation of the QCR\_SECRET and the IKE SPI as follows:

$$\text{TOKEN\_SECRET\_DATA} = \text{HASH}(\text{QCR\_SECRET} \mid \text{SPI-I} \mid \text{SPI-R})$$

- o Verification uses the same calculation, and works even if the IKE SA has been deleted. Still, if the IKE SA is no longer valid, the notification MUST NOT be acknowledged, as this could be used in an attempt to guess the QCR\_SECRET.

### [5.3.](#) Token Lifetime

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

## [6.](#) Alternative Solutions

### [6.1.](#) Why not Save the Entire IKE SA

IKEv2 does not assume the existence of a persistent storage module. If we are adding such a module, why not use it to save the entire IKE SA across reboots, nullifying the need for a crash recovery procedure?

There are several reasons why we believe that this is not a good idea:

1. A token is only 16-256 octets, and is much more compact than all the data needed to store an IKE SA.
2. A token is valid for the life of an IKE SA. An IKE SA state is updated whenever a message is sent, because of the requirement to keep the sequence of message IDs. It may not be acceptable to update the persistent storage whenever an IKE message is sent.
3. A reboot is usually an unpredictable event, and as such, we cannot know how long it will last. By the time the machine has rebooted, the peer may have attempted some type of protected exchange (liveness check, create-child-SA or delete), timed out, and deleted the SA. It is far better to reboot without SAs and

with only a token for quick recovery.

## [6.2.](#) Initiating a new IKE SA

Instead of sending a QCR 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 assymetric, such as when EAP is used, it is not possible for the rebooted implementation to initiate IKE.

## [7.](#) Operational Considerations

To support this standard, an implementation needs to have access to a persistent storage module. This could be an internal hard disk, a local or remote database application, or any other method that persists across reboots. This storage module and the data links between the storage module and the IKE module must meet the performance requirements of the IKE module. The storage module **MUST** support insertion and deletion rates equal to peak IKE SA setup rates and it **SHOULD** support query rates that are fast enough.

See [Section 8](#) for security considerations for this storage mechanism.

In order to limit the effects of DoS attacks, an implementation **SHOULD** limit the rate of queries into the token storage so as not to overload it. If excessive amounts of IKE requests protected with unknown IKE SPIs arrive, 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.

## [8.](#) Security Considerations

Tokens **MUST** be hard to guess. This is critical, because if an attacker can guess the token associated with the 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 back in an informational exchange it is not encrypted, but that is

the last use of that token.

An aggregation of some tokens generated by one peer together with the related IKE SPIs MUST NOT give an attacker the ability to guess other tokens. Specifically, if one peer does not properly secure the QCR tokens and an attacker gains access to them, this attacker MUST NOT be able to guess other tokens generated by the same peer. This is the reason that the QCR\_SECRET in [Section 5.2](#) needs to be long.

The persistent storage MUST be protected from access by other parties. Anyone gaining access to the contents of the storage will be able to delete all the IKE SAs described in it.

The tokens associated with expired and deleted IKE SAs MUST be deleted from the storage, so that a future compromise of the storage does not reveal enough tokens to facilitate an attack against the QCR tokens.

The QCR 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.](#) IANA Considerations

IANA is requested to assign a notify message type from the error types range (43-8191) of the "IKEv2 Notify Message Types" registry with name "QUICK\_CRASH\_RECOVERY".

## [10.](#) 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.

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

[resumption]  
Sheffer, Y., Tschofenig, H., Dondeti, L., and V.

Nir

Expires September 17, 2008

[Page 10]

---

Internet-Draft

Quick Crash Recovery

March 2008

Narayanan, "IPsec Gateway Failover Protocol",  
[draft-sheffer-ipsec-failover-02](#) (work in progress),  
November 2007.

#### Author's Address

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

Email: [ynir@checkpoint.com](mailto:ynir@checkpoint.com)

## Full Copyright Statement

Copyright (C) The IETF Trust (2008).

This document is subject to the rights, licenses and restrictions contained in [BCP 78](#), and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY, THE IETF TRUST AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

## Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information

on the procedures with respect to rights in RFC documents can be found in [BCP 78](#) and [BCP 79](#).

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at <http://www.ietf.org/ipr>.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at [ietf-ipr@ietf.org](mailto:ietf-ipr@ietf.org).

#### Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).