Internet Engineering Task Force
IP Security Working Group
Internet Draft

Andrew Krywaniuk
Alcatel Networks Corporation
T. Kivinen
SSH Communications Security
July 14, 2000

Using Isakmp Heartbeats for Dead Peer Detection
<draft-ietf-ipsec-heartbeats-01.txt>

Status of this Memo

This document is a submission to the IETF Internet Protocol Security (IPsec) Working Group. Comments are solicited and should be addressed to the working group mailing list (ipsec@lists.tislabs.com) or to the editor.

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC 2026.

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 made obsolete 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.

Copyright Notice

Abstract

This document describes a method for sending heartbeat packets (sometimes called keep-alives) across an ISAKMP SA in order to detect when a peer has crashed or has become otherwise unreachable. A method for negotiating the heartbeat parameters is also provided.

Table of Contents

1. Introduction

Heartbeat packets have often been used (particularly at the link layer) to detect when a peer has crashed (hung up, etc.) in order that the necessary corrective or cleanup action can be performed.

When the link is secured using the IPsec protocol suite, special precautions must be taken in order to ensure that the heartbeat packets are also sent in a secure manner.

This document describes a method for negotiating and implementing a heartbeat protocol that runs on top of ISAKMP. This protocol prevents an adversary from generating false proof of liveness/deadness in a manner that is resistant to a variety of DoS attacks.

2. Specification of Requirements

This document shall use the keywords "MUST", "MUST NOT", "REQUIRED","SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED, "MAY", and "OPTIONAL" to describe requirements. They are to be interpreted as described in [Bradner97].

3. Changes Since Last Revision

The document has been reorganized based on comments since the initial revision. Protocol specifications have been moved closer to the beginning of the document; background information and implementation hints have been moved closer to the end. The details of the protocol have not been significantly altered, due to a lack of agreement among WG members as to what, if any, changes are required.

4. Document Roadmap

Sections 5-10 provide the technical details of the protocol.

Section 11 provides background information regarding the protocol design. It elucidates the goals of the protocol and it explains how the protocol can be extended.

Section 12 provides a variety of implementation hints. Many of these will aid in interoperability with existing IPsec implementations.

5. Terminology Used Throughout This Document

The term 'keep-alive' literally refers to an exchange which keeps a connection open by periodically exchanging packets with the peer. Over time, the term has also come to refer to an exchange which detects if the connection is still open. If this detection mechanism can trigger an action which will restore the connection when it goes down then it is still, in effect, functioning as a keep-alive.

This document uses the term 'heartbeat' to refer to a category of periodic keep-alive packets which can be used to determine the current liveness or reachability of the peer (in the same way that a doctor might use a heartbeat to determine if a patient is alive or dead). However, a heartbeat does not attempt to keep the connection open by defeating the peer's inactivity timeout mechanism.

A 'proof of liveness' is a payload or message which indicates that the peer is still up and running, and is capable of sending and receiving traffic on the given ISAKMP SA.

A 'dead peer' is a peer that has failed, rebooted, or is otherwise unable to communicate with the host using the ISAKMP SA. For convenience's sake, we include the case where the connection between the peers has gone down (e.g. the user hung-up, there was a routing error, the ISAKMP SA was deleted, etc).

The 'sender' is the term that is used to identify the host who sends the heartbeat packets. Similarly, the 'receiver' is the host who receives the heartbeats.

Throughout this memo, the 'initiator' refers to the host who initiated the heartbeat negotiation (not the initiator of the ISAKMP SA). The term 'responder' is interpreted likewise. Note that currently the initiator will always be the receiver of heartbeats and the responder will always be the sender.

6. Basic Packet Format

The heartbeat exchange is unidirectional. In other words, it is a one packet exchange.

The format of the packet looks like this:

```
    Sender                                  Receiver
  -----------                              -----------
    HDR*, SEQ_NO, HASH,
    NOTIFY(Still Connected)     -->
    [,EXTRA PAYLOADS]
```

An implementation MUST use the above payload order.

The Exchange Type field in the ISAKMP header MUST be set to HEARTBEAT_MODE. HEARTBEAT_MODE has been assigned the value 251 from the private range.

The SEQ_NO payload allows the receiver to discard packets that may have been spoofed or replayed. It is described in section 6.1.

The HASH payload is described in section 6.2.

The NOTIFY(Still Connected) payload is what actually provides the liveness proof. It is described in section 6.3.

A host MAY choose to add extra payloads to the end of the message. However, these payloads SHOULD be ignored unless they have been enabled via the Heartbeat Negotiation Protocol or by a vendor-specific extension mechanism.

6.1 The SEQ_NO Payload

The SEQ_NO payload is a new ISAKMP payload with value 217 (from the private range).

It has this format:

```
                     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  !   RESERVED    !         Payload Length        !
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  !                       Sequence Number                         !
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The sequence number MUST be 32 bits long unless otherwise negotiated (no negotiation mechanism is currently provided). This implies that the payload length will normally be 8 bytes.

6.2 The HASH Payload

The HASH is calculated according to the recommendations in [REVISED-HASH].

HASH = prf(SKEYID_a, PAYLOAD_FRAG_1 | HASH_0 | PAYLOAD_FRAG_2)

Where:

PAYLOAD_FRAG_1 is the set of ISAKMP payloads that precede the HASH.
HASH_0 is a HASH payload with an empty hash (all 0s).
PAYLOAD_FRAG_2 is the set of ISAKMP payloads that follow the HASH.

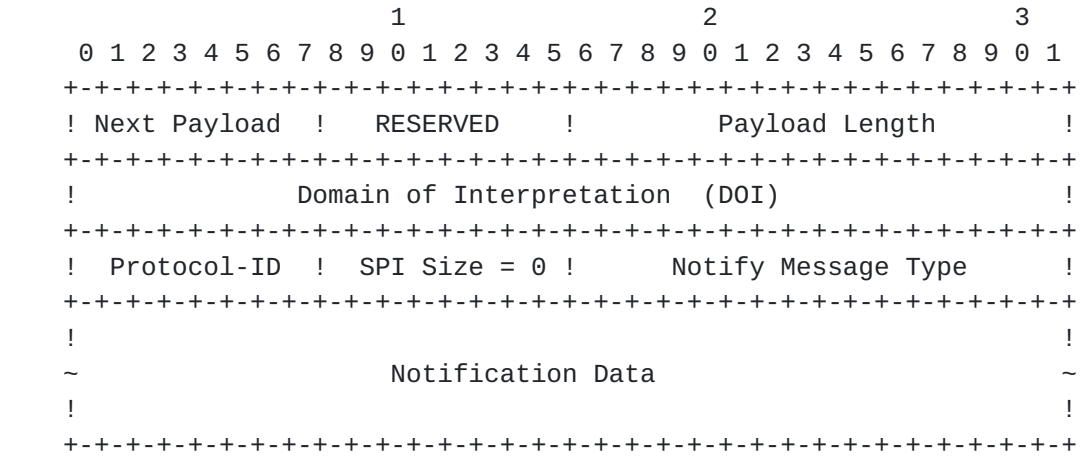
In the typical case:

PAYLOAD_FRAG_1 = HDR | SEQ_NO
HASH_0 = Payload Header | sizeof(HASH) bytes of 0
PAYLOAD_FRAG_1 = NOTIFY(Still Connected) [| SPI_LIST(s)]

6.3 The NOTIFY(Still Connected) Payload

The NOTIFY(Still Connected) payload is a standard ISAKMP Notify payload with a new notify type, STILL-CONNECTED. The value of the new notify type is 34793 (from the private range for status notifies)

The format of the notify is shown below:



The DOI for this notify SHOULD normally be IPsec. The Protocol-ID SHOULD be PROTO_ISAKMP. The optional SPI field SHOULD be omitted and the SPI size SHOULD therefore be set to 0.

The sequence number MAY be included in the Notification Data. However, since parsing of the Notification Data is not required, the SEQ_NO payload at the top of the packet is the master copy.

7. The Heartbeat Protocol

Section 6 described the format of the heartbeat packets. This section describes the processing required to handle the packets.

[The text in this section assumes that the sequence number will always be 32 bits. If this limit is increased in the future, the text will be modified to reflect the new value.]

7.1 Sending Heartbeat Packets

After agreeing to use heartbeats (using the Heartbeat Negotiation Protocol or some other mechanism), the sender MUST begin sending heartbeat packets at the negotiated interval.

Each packet contains a sequence number, which is incremented for successive packets. Note that the sender MUST increment the initial sequence number BEFORE sending the first heartbeat packet.

The sequence number is not allowed to wrap from 0xffffffff to 0. If this does happen then the heartbeats MUST be stopped and the sender SHOULD attempt to rekey the ISAKMP SA. To avoid this possibility, choose a sequence number that is less than 2^31.

7.2 Receiving Heartbeat Packets

Because there may be race conditions due to setup times, the receiver SHOULD begin listening for heartbeats as soon as possible after the negotiation completes.

Each time a heartbeat packet arrives, the receiver must verify the hash information and ensure that the sequence number falls within an acceptable window. If either one of these two conditions fails, the packet is invalid and should be discarded.

If the packet passes these tests then the receiver must update his last-known-good sequence number variable to the value contained in the packet. Also, he should maintain a state for the time at which the last valid heartbeat was received.

7.3 Receiver Background Tasks

The receiver must periodically check the stored heartbeat state in order to verify that the timeout interval has not elapsed without the reception of a valid heartbeat packet from the peer.

If this happens then the ISAKMP SA and any corresponding IPsec SAs SHOULD be deleted. Delete notifications MAY be sent, but they will presumably not be understood by the peer, since the connection is verifiably dead. The receiver MAY also attempt to rekey the SA.

The receiver SHOULD also periodically check for time slippage (the sequence number should increase at a rate proportional to the elapsed time).

8. Heartbeat Negotiation Protocol

In order to promote interoperability, this memo also includes a standardized method of negotiating heartbeat parameters.

Of the various parameters, the only one that MUST be agreed upon by the two hosts is the heartbeat interval. However, we also use the negotiation protocol to indicate support for optional payloads, such as the SPI_LIST.

In general, the sender (responder) has the final decision regarding the heartbeat parameters. The initiator may propose values for the heartbeat options and heartbeat interval in the Config Request, but the responder MAY ignore these values where it makes sense to do so.

8.1 Negotiation Transport

The negotiation of heartbeat parameters can be accomplished using the technique described in the ISAKMP Configuration Method [IKECFG]. However, the implementer is NOT REQUIRED to use ISAKMP-Config for other purposes, such as assigning internal network addresses.

ISAKMP-Config may be replaced with a similar generic attribute exchange protocol in the future. If support for an alternate protocol can be verified (e.g. by a vendor id) then the initiator may use that protocol instead.

8.2 Heartbeat Configuration Attributes

We define the following new configuration attributes (taken from the private range):

```
HEARTBEAT_TYPE = 22565
HEARTBEAT_OPTIONS = 22566
HEARTBEAT_INTERVAL = 22567
```

```
  HEARTBEAT_PROPOSAL_ACCEPTED = 22568
  SEQUENCE_NUMBER = 22569
```

An implementation using the Heartbeat Negotiation Protocol MUST recognize all of these attributes.

The attributes should be interpreted as follows:

a) HEARTBEAT_TYPE (4 bytes, enum)

Value	Meaning
0	RESERVED
1	Standard
2-32767	Reserved for future use
32768-65535	Reserved for private use

The HEARTBEAT_TYPE will be used to upgrade the heartbeat protocol in the future. New versions of this document may deprecate the current standard heartbeat type by defining a new value from the future use range.

Consenting parties (as defined in [EXT-METH]) may choose to negotiate a completely different heartbeat mechanism by using values from the private use range.

This attribute MUST be included in every heartbeat negotiation packet (and it SHOULD be the first attribute in the list). This enables the peer to quickly identify an ISAKMP-Config packet as part of a heartbeat negotiation.

b) HEARTBEAT_OPTIONS (4 bytes, bitfield)

Value	Meaning
0x00000001	Support SPI_LIST
0x00000002	Authentication Only
0xFFFFFFFC	Reserved for future use

No private usage range is defined. Consenting parties wishing to negotiate private options MAY define a new ISAKMP-Config attribute.

An implementation SHOULD ignore any proposed options which it does not understand. Reception of an unrecognized option SHOULD NOT cause the negotiation to fail.

The Authentication Only option may be deprecated by a future version of this draft (in a backwards-compatible manner) if the working group

decides that encryption MUST be either on or off for heartbeat packets.

c) HEARTBEAT_INTERVAL (4 bytes, # of seconds)

This attribute MUST be negotiated by the peers. If the sender and receiver do not use the same heartbeat interval, the heartbeat protocol will not work properly.

d) HEARTBEAT_PROPOSAL_ACCEPTED (4 bytes, Boolean)

This attribute confirms that the peer has agreed to do heartbeats with the negotiated parameters.

Value	Meaning
0	Rejected
1	Accepted
2-65535	RESERVED

To maintain compatibility with future versions of this document, an implementation SHOULD normally send HEARTBEAT_PROPOSAL_ACCEPTED=0 when rejecting a heartbeat proposal. The inclusion of the Proposal Rejected attribute indicates to the initiator that he SHOULD NOT retry the negotiation.

e) SEQUENCE_NUMBER (variable size (4 bytes is standard), int)

This is the initial sequence number, which is used as a seed for the responder's LKG_SN. It SHOULD be chosen randomly from the range [0, 2^31].

8.3 Sample Heartbeat Negotiations

Example of a simple configuration exchange:

```
  Initiator                                              Responder
  --------------                                    -----------------
  REQUEST(HEARTBEAT_TYPE=Standard)  -->

                                    <--  REPLY(HEARTBEAT_TYPE=Standard,
                                               HEARTBEAT_INTERVAL=30,
                                               SEQUENCE_NUMBER=1234,
                                         HEARTBEAT_PROPOSAL_ACCEPTED=1)
```

Example where the initiator proposes some values:

```
  Initiator                                                Responder
  --------------                                      -----------------
  REQUEST(HEARTBEAT_TYPE=Standard,
          HEARTBEAT_INTERVAL=20,
          HEARTBEAT_OPTIONS=Send SPI list)  -->

                                    <--  REPLY(HEARTBEAT_TYPE=Standard,
                                                 HEARTBEAT_INTERVAL=30,
                                            HEARTBEAT_OPTIONS=Send SPI list,
                                                 SEQUENCE_NUMBER=1234,
                                           HEARTBEAT_PROPOSAL_ACCEPTED=1)
```

Example where the responder doesn't want to send heartbeats:

```
  Initiator                                                Responder
  --------------                                      -----------------
  REQUEST(HEARTBEAT_TYPE=Standard)  -->
                                    <--  REPLY(HEARTBEAT_TYPE=Standard,
                                           HEARTBEAT_PROPOSAL_ACCEPTED=0)
```

Example where the initiator is using a newer version of the heartbeat protocol:

```
  Initiator                                                Responder
  --------------                                      -----------------
  REQUEST(HEARTBEAT_TYPE=2)  -->
                                         <--  REPLY(HEARTBEAT_TYPE=1)
  REQUEST(HEARTBEAT_TYPE=1)  -->

                                         <--  REPLY(HEARTBEAT_TYPE=1,
                                                HEARTBEAT_INTERVAL=30,
                                                 SEQUENCE_NUMBER=1234,
                                        HEARTBEAT_PROPOSAL_ACCEPTED=1)
```

9. The SPI_LIST Payload

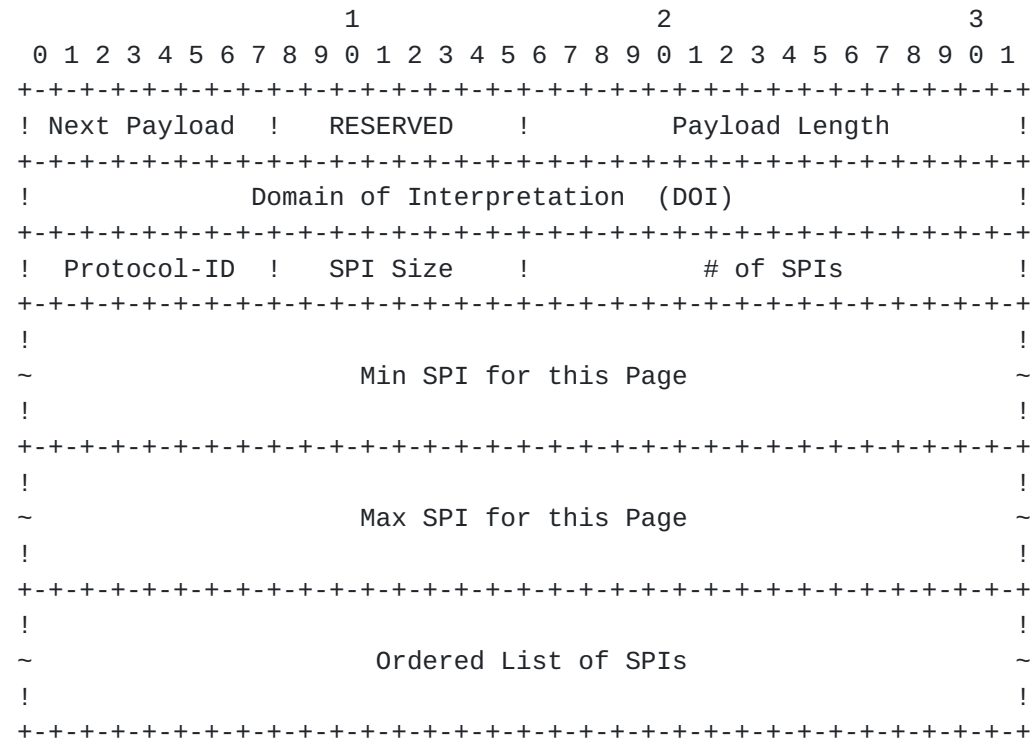
The SPI list is an optional payload which allows the peers to synchronize SAD information during the heartbeat exchange. In conjunction with the basic Heartbeat Protocol, this allows an IPsec host to be more fully self-stabilizing.

9.1 Payload Format

The SPI_LIST payload is a new ISAKMP payload with value 218 (from the private range). It provides information about which SPIs are currently known to the peer.

Its format is similar to that of the ISAKMP Delete payload, but its function is almost the exact opposite. Whereas the delete notification tells the peer that all SPIs on the list are no longer valid, the SPI list tells the peer that any SPIs NOT on the list are no longer valid.

The format of the SPI_LIST payload is as follows:



The DOI for this payload should normally be IPsec. The Protocol-ID SHOULD be either PROTO_IPSEC_ESP or PROTO_IPSEC_AH, so the SPI Size would normally be 4 bytes. The SPI list MAY also be used to track IPCOMP SAs, in which case the SPI size would be 2 bytes.

The SPI Size determines the width of the Min SPI and Max SPI fields. It also determines the width of each entry in the ordered list. The 'Number of SPIs' field only refers to the number of SPIs in the ordered list; the Min SPI and Max SPI fields are not counted.

The ordered list MUST ONLY include outbound SPIs (relative to the sender). The reason for this is explained below. Also, the list MUST always be sorted in ascending order.

9.2 Using the SPI list

The SPI list may be included in the "Optional Payloads" section of the Heartbeat packet. Implementations are NOT REQUIRED to parse this payload (but they must recognize and ignore it).

In cases where the number of SAs between a pair of host is small (the normal case), it may be acceptable to include all of the SPIs inside a single SPI_LIST payload. In this case, the Min and Max SPI field SHOULD be set to 0 and 0xFFFFFFFF respectively (for 32 bit SPIs).

If, however, there is a large number of IPsec SAs between the two hosts, it may be desirable to split the list of SPIs into multiple 'pages'. In this case, the ordered list MUST ONLY contain those SPIs which fall within [Min SPI, Max SPI]. See section 12.8 for some suggestions on how to split the list into pages.

Also, a SPI_LIST payload is bound to only one Protocol and DOI. To send SPI lists for multiple protocols, multiple SPI_LIST payloads are required.

The receiver of the SPI List SHOULD search his SA database(s) for inbound SPIs matching the criteria of (DOI, Protocol, [Min SPI, Max SPI]). If the peers are correctly synchronized, this list should match the outbound SPI list from the heartbeat packet.

If, however, the two lists differ, then some corrective action SHOULD be taken. If an SA from the packet is missing from the SAD then a delete notification SHOULD be sent. If an SA from the SAD is missing from the packet then the SA should be deleted from the SAD.

10. Use of Values from the Private Range

This memo describes a protocol that is still in the experimental stages. As such, all protocol-specific constants have been assigned from the private range.

Use of these constants is enabled by the exchange of the following vendor id:

Vendor Id = 0x8DB7A41811221660

If and when this document is accepted by the IETF for incorporation into the set of IPsec standards, all or some of the following will occur:

The heartbeat exchange mode, SEQ_NO payload, SPI_LIST payload, STILL-CONNECTED notify type, and all the ISAKMP-Config attributes will be assigned permanent values by the IANA or the editors of the relevant drafts.

The description of the SEQ_NO and SPI_LIST payloads will be added to [ISAKMP].

The descriptions of the ISAKMP-Config attributes will be added to [IKECFG].

Text will be added to [NOTIFY-DATA] to describe the additional data section of the STILL-CONNECTED notify.

The vendor id will no longer be needed.

11. General Approach

The conceptual idea behind the heartbeat protocol is simple:

A host wishes to detect, in a timely fashion, when a peer has crashed or is otherwise unreachable. In order to accomplish this, he asks the peer to send him periodic 'heartbeat' packets on a secure connection. If the heartbeats stop, then the connection is no longer valid and some corrective or diagnostic action should be taken.

11.1 Security Considerations

When the heartbeats mechanism is being used in conjunction with a security protocol, there are a few additional wrinkles to be considered:

Firstly, an adversary must not be able to provide false proof of liveness by replaying old packets. This implies that the packet must include some kind of shared state which proves to the recipient that it was generated recently.

Since it would be over-ambitious to assume that the system time is synchronized with GMT on every host, we do not rely on a timestamp to provide the liveness proof. Instead, we use a monotonically increasing sequence number. If the adversary replays an old packet, the peer will detect the old sequence number and he will reject the packet.

Secondly, an adversary must not be able to provide false proof of liveness by spoofing packets. Therefore, each packet must be authenticated using a keyed hash. This is accomplished by sending heartbeats under the protection of an existing ISAKMP SA. If the adversary spoofs a packet, the peer will detect the invalid hash information and he will reject the packet.

11.2 Goals of the Heartbeat Protocol

This protocol was designed with certain specific goals in mind.

This version of the protocol aims satisfy all of the primary goals and as many of the secondary goals as possible without sacrificing simplicity.

The future considerations section (Appendix A) suggests extensions that may be used in the future in order to satisfy more of the secondary goals. These extensions will be evaluated based on comments from vendors who have implemented working prototypes of the protocol.

The primary goals of the heartbeat protocol are:

a) To provide a simple dead peer detection protocol that is easy to implement.
b) To not weaken the security of existing IPsec protocols.
c) To promote interoperability between vendors by using an unambiguous packet format.

The secondary goals of the heartbeat protocol are:

d) To ensure that the protocol is not subject to packet spoofing, replay, or flooding attacks.
e) To detect the dead peer in as timely a manner as possible.

f) To detect missing IPsec SAs (due, perhaps, to lost deletes or to crashed IPsec devices).
g) To provide a flexible negotiation scheme for the heartbeat protocol.
h) To ensure that neither of the two parties is overloaded by the heartbeat packets.

11.3 Design Considerations

In order to make the heartbeat protocol more modular, we have separated the design into three layers, where each layer is only dependent on the layer directly below it.

This means that the heartbeat protocol could be adapted to other situations. Implementers wishing to use one of the other possible heartbeat types (see Appendix B) could redefine layers 1 and 2 (by using a different heartbeat type during negotiation) or layer 3 (by sending a different vendor id during phase 1 negotiation).

LAYER 1 (Content): At the lowest level, we define a proof of liveness payload (Notify Still Connected). This payload provides proof of liveness whenever it is transmitted over a secure channel along with a valid sequence number.

LAYER 2 (Transport): As a standard method of delivering the liveness proof, we define a heartbeat exchange mode. The heartbeat exchange uses the security of an existing ISAKMP SA to transport the Notify Still Connected payload and the sequence number.

LAYER 3 (Negotiation): In order to negotiate the use of the heartbeat exchange, we define a standard heartbeat negotiation protocol. This protocol uses ISAKMP-Config to communicate important parameters and options to the peer.

The proof of liveness payload we have chosen is a new notify type called Notify Still Connected. Reception of this Notify payload (with a valid sequence number and on a secure connection) is enough to guarantee that the peer is alive and that the connection is still valid.

Of course, there is no corresponding 'proof of deadness' payload; a peer that is dead will generally be unable to send such a payload (at least not in a secure manner). Proof of deadness is achieved when the peer fails to provide proof of liveness within a given timeout interval.

The heartbeat exchange provides a standard transport mechanism for the notify payloads. It ensures reliable delivery, protects against

some kinds of DoS attacks and provides additional features, such as the ability to recover from lost ISAKMP Delete notifications or to detect crashed IPsec devices.

The standard way of negotiating to use the heartbeat exchange is via the heartbeat negotiation protocol. This protocol allows the peers to agree on important parameters, such as the frequency with which heartbeat packets are sent, and support for optional payloads.

The negotiation protocol also provides a mechanism for modifying or extending the heartbeat protocol in the future.

12. Implementation Hints

Although, the Heartbeat framework is fairly generic, we expect that most implementations will choose the same basic approach.

For example, a logical constraint for dead peer detection is fixed worst case response. By controlling various implementation constants, we can ensure that a dead peer is always detected within a given timeout interval.

12.1 Terminology Used in This Section

The 'heartbeat interval' (HB_I) is the negotiated rate (in seconds per packet) at which the sender has agreed to send heartbeat packets.

The 'timeout interval' (TO_I) is the elapsed time (in seconds) after which the receiver should consider the sender to be dead (if no heartbeat is received during this period).

The 'lost packet tolerance' (LP_T) is the number of heartbeats that must be lost before the receiver should consider the sender to be dead.

The 'packet transmission window' (PT_W) is the maximum reasonable time (in seconds) that it should take for a packet to be generated by the sender, transmitted across the Internet, and processed by the receiver.

The logical relationship between these values SHOULD be as follows:

```
TO_I = HB_I x LP_T + PT_W
SN_W = LP_T + 1
```

The 'last known good sequence number' (LKG_SN) is the stored value of the sequence number from the last heartbeat packet from the peer that passed all authentication and validity checks.

The 'initial sequence number' (SN_0) is the pre-negotiated sequence number which is used as the original value of LKG_SN.

The 'sequence number window' (SN_W) is the maximum acceptable jump in the sequence number between consecutive valid heartbeat packets. The receiver should discard any incoming packets when the sequence number does not fall within the range of [LKG_SN + 1, LKG_SN + SN_W].

The 'time slippage window' (TS_W) is the maximum acceptable discrepancy between the current sequence number and the current time. After N heartbeat packets have been sent, the sequence number should have increased by N and the time should have increased by HB_I x N.

If (elapsed time) - HB_I x (LKG_SN - SN_0) > TS_W then you have possible evidence of tampering by an intermediate router.

12.2 Suggested Values for Heartbeat Parameters

Choosing the values for the various heartbeat parameters is, for the most part, a matter of local policy. However, here we present a list of constraints and suggested values for these parameters.

A suggested value for the lost packet tolerance is 3. It seems unlikely that, under normal circumstances, three consecutive packets would be lost (especially when they are spaced out at regular intervals).

A suggested value for the packet transmission window is 5 seconds. This includes a fairly substantial safety margin.

A suggested value for the heartbeat interval is 20 seconds. Using the standard formula (TO_I = HB_I x LP_T + PT_W), this implies that the suggested value of the timeout interval will be 65 seconds.

Some possible methods for reducing the timeout interval in the future are discussed in Appendix A.

Using these values, we expect that heartbeat packets will not place

undue load on either the sender or the receiver. Assuming an average of 100 bytes per heartbeat packet, heartbeat packets will amount to only 5 bytes of traffic per second per channel.

In the case of a large deployment, a host that is sending or receiving heartbeat traffic on 1000 simultaneous channels will only be burdened with 5kb/s (50 packets/s) of extra traffic.

A suggested value for the time slippage window is 200 seconds. The value of this parameter is not critical, but it SHOULD be greater than TO_I. Also, the upper bound on this parameter SHOULD be set relative to the ISAKMP SA lifetime (e.g. it should not be more than 10% of the SA lifetime).

12.3 Optimizing for Speed

Since the heartbeat protocol is unidirectional and not reliant on any interaction with the peer, heartbeat packets may be built in advance (during spare CPU cycles) and then stored until they are scheduled to be sent.

A host MAY choose to use unencrypted heartbeat packets. However, he may only do so if this option has been specifically negotiated with the peer.

Unencrypted heartbeats provide faster throughput in the normal case, but encrypted packets may provide faster rejection of spoofed packets unless some other DoS resistance technique is being used (see Appendix A). The security ramifications of using unencrypted packets are discussed in the Security Considerations section.

12.4 Optimizing for Reliability

The sender MUST attempt to send the packet within a short window of the heartbeat interval. If the receiver does not consistently receive the packet within PT_W of the negotiated interval then the reliability of the heartbeat protocol will be diminished, since the lost packet tolerance will effectively be reduced by 1.

The receiver SHOULD also periodically check that the time slippage window has not been exceeded. If this check fails, it may indicate that an intermediate router is storing packets and delaying their transmission in order to setup a future false proof of liveness.

When the adversary is an intermediate router, little can be done to ensure the reliable and timely delivery of packets. One possible

remedy is to send the heartbeat packets with the Type of Service field set to high reliability. This increases the probability that some heartbeat packets will manage to avoid passing through the malicious router.

12.5 Optimizing for State

It is theoretically possible for the sender of the heartbeat packets to operate in an essentially stateless manner.

To do this, the sender must always choose the same heartbeat interval and he must keep a global sequence number state.

Although it is recommended that the responder choose a heartbeat interval that is no less than the one the initiator proposed, the stateless heartbeat sender MAY break this rule.

In this case, the receiver MAY compensate by choosing to only parse every Nth heartbeat packet. To do this, he SHOULD adjust his normal heartbeat parameters as follows:

```
HB_I = HB_I x N
LP_T = LP_T x N
SN_W = SN_W x N
```

12.6 Filtering Heartbeat Packets

The receiver may use a variety of mechanisms in order to speed up his rejection of invalid heartbeat packets (thereby reducing his vulnerability to DoS attacks). Use of these filtering techniques is NOT REQUIRED.

He MAY ignore heartbeat packets that arrive when a heartbeat is not expected (i.e. within HB_I - PT_W of the last valid heartbeat).

He MAY ignore a packet if the NEXT_PAYLOAD field in the ISAKMP Header is not set to SEQ_NO.

He MAY ignore a packet after decrypting the first block if the sequence number is out of range.

He SHOULD check that the encryption bit in the ISAKMP Header is off if he has negotiated to use authentication only.

Other possible filtering mechanisms are suggested in Appendix A.

12.7 Managing the Sequence Number

For security reasons, the sequence number must not be allowed to cycle through all 2^32 possible values. This would allow an adversary to successfully replay an old, stored packet.

For practical reasons, we do not allow the sequence number to wrap from 0xffffffff to 0, since this would require added complexity in the algorithm that checks the sequence window.

A suggested algorithm for generating the initial sequence number is to choose a random 32 bit number and then set the high bit to 0. This ensures that at least 2^31 heartbeat packets can be sent on the ISAKMP SA.

12.8 Use of the SPI List

Generally speaking, reception of the Still Connected Notify only provides proof that the peer's ISAKMP process is still up and running. In order to provide a finer granularity of dead peer detection, the SPI list can be used to ensure that the SADs of the two peers also remain synchronized.

This may be useful when the ISAKMP process is running on a different machine from the IPsec process(es). It may be possible for one or more of the IPsec devices to crash or otherwise delete its SAs, even though the ISAKMP process continues to send valid heartbeat packets.

If the SPI List is used, the ISAKMP process SHOULD periodically query each IPsec process in order to verify that it is still working correctly and that local cached copies of the SAD are properly synchronized.

Since support for the SPI_LIST payload is optional, it should not be used unless the peer has indicated support for it (via the Heartbeat Negotiation protocol or by some other mechanism).

Depending on the number of IPsec SAs that exist between the two peers, the complete SPI list(s) may grow quite large and it may not be desirable to include all the SPIs in every heartbeat packet. Therefore, one or more of the following approaches is suggested:

a) Include the SPI list only occasionally (e.g. in every Nth packet).
b) Split the SPI list into N equal 'pages' and send one page in each packet (this requires a stored state).
c) For each packet, generate a random Min SPI and use it to choose

a random page of fixed size (this requires no extra stored state).

d) Send the SPI list when the ISAKMP process detects that one of the IPsec devices has crashed.

e) Send the SPI list after receiving a large number of packets with invalid SPIs.

Implementation Hint:

When sending a page of SPIs, don't just set the Min and Max SPI variables to the first and last entries in the ordered list. The purpose of the SPI list is to indicate which SPIs are not being used; therefore, the range of SPI values should be as wide as possible.

Note that the reason why the SPI_LIST lists the sender's outbound SPIs is that the receiver may need to send a delete notification. If the SPI_LIST had used the sender's inbound SPIs (receiver's outbound) then the receiver might have been unable to correlate the invalid outbound SPI with the appropriate invalid inbound SPI.

If the missing SPI is part of an SA bundle (as defined in [ARCH]), it may be permissible for the receiver to delete the entire bundle. However, this SHOULD NOT be done unless the peer has indicated support for this behaviour (e.g. through a private heartbeat option).

It is unclear what an implementation should do if a reserved SPI value (e.g. 0-255) is included in the SPI_LIST. Documents which allocate SPI values from the reserved range SHOULD specify this behaviour. If no specific behaviour is specified then these SPI values SHOULD be ignored.

12.9 Rules for Negotiation

In general, the sender (responder) has the final decision regarding the heartbeat parameters. The initiator may propose values for the heartbeat options and heartbeat interval in the Config Request, but the responder MAY ignore these values where it makes sense to do so.

If the initiator proposes a value for the heartbeat interval, the responder SHOULD normally either accept that value or choose a longer interval (slower frequency).

If the initiator does not propose to use the SPI List, the responder SHOULD choose NOT to send it. There is no value in sending the SPI List if the receiver has indicated that he will not parse it.

Support for the authentication only mode of heartbeat is NOT

REQUIRED. Therefore, if the initiator does not propose this mode, the responder MUST NOT use it.

If the initiator proposes a heartbeat type from the private or future use ranges (i.e. the initiator is using a different version of the Heartbeat Negotiation Proposal), the responder SHOULD respond by setting the HEARTBEAT_TYPE to his standard value, but he SHOULD NOT send HEARTBEAT_PROPOSAL_ACCEPTED=0. This indicates that the initiator SHOULD retry the negotiation using the responder's preferred heartbeat type (if he supports it).

The Heartbeat Negotiation Protocol only negotiates unidirectional heartbeats. If both peers wish to receive heartbeats, they should each initiate heartbeat negotiation exchanges (the two exchanges will be independent of each other).

The negotiated heartbeat protocol is bound to an ISAKMP SA. If the SA is rekeyed, the heartbeat protocol SHOULD be renegotiated using the new ISAKMP SA. If there is more than one ISAKMP SA between the peers, it is not necessary to send heartbeats on both of them.

The heartbeat negotiation process is currently not replay resistant. Therefore, once heartbeats have been successfully negotiated with a peer, the implementation MUST ignore all subsequent heartbeat requests on the same phase 1 SA.

Possible extensions to the protocol to make the negotiation process replay resistant are suggested in Appendix A.

12.10 Dealing with Dangling SAs

Some implementations have been categorized as 'dangling SA' hosts. This means that they will delete Isakmp SAs under some conditions (e.g. low memory) when corresponding IPsec SAs still exist. This behaviour has been deemed acceptable by the IPsec Working Group, and therefore it MUST be supported.

Although heartbeats cannot be sent under these conditions, the heartbeat protocol has been specifically designed to ensure that termination of the heartbeats will not cause the peer to delete the IPsec SAs.

When either peer deletes the ISAKMP SA, the heartbeats MUST be stopped. Therefore, it is imperative that the delete notification be sent over a reliable delivery mechanism. Use of the Acknowledged Informational exchange (see [IKEv2]) for this purpose is encouraged.

In the case where the receiver of the heartbeats sends the delete using an unacknowledged notify message, he SHOULD store the delete notification for a limited time and periodically retransmit it if he continues to receive heartbeat traffic on the deleted SA.

12.11 Dependence on ISAKMP-Config

If an implementation wishes to use ISAKMP-Config to transport the Heartbeat Negotiation Protocol, that implementation MUST implement the basic framework for sending and receiving ISAKMP-Config messages.

According to the text of [IKECFG], an implementation is REQUIRED to recognize all mandatory ISAKMP-Config attributes (e.g. INTERNAL_IP4_ADDRESS, APPLICATION_VERSION).

However, actual support for these features is not required. If the host does not implement full support for the attribute sent in the CONFIG-REQUEST, he may indicate that the option is not available by simply removing the attribute from the CONFIG-REPLY.

13. Security Considerations

The focus of this document is security; hence security considerations permeate this specification.

This document discusses a method of sending heartbeat traffic across a secure channel. Use of an insecure heartbeat protocol would allow an adversary to provide false proof of liveness.

The ability to provide false proof of liveness might assist the peer in performing a DoS attack or in preventing an implementation from minimizing the damage done when a key is compromised.

Also, in the absence of a standardized dead peer detection protocol, an implementer might be tempted to rely on insecure mechanisms, such as unauthenticated INVALID-SPI or INVALID-COOKIE notifications, which can be used to provide false proof of deadness.

Implementations which use the authentication only mode of heartbeats should be aware that an adversary will have access to the SPI list and to a large number of known-plaintext hash outputs. The use of encryption guarantees an equivalent level of security to Quick Mode or other phase 2 [IKE] modes.

14. Acknowledgments

The authors would like to thank the members of the IPsec working group who contributed ideas for the design of this protocol, especially Jan Vilhuber, Bronislav Kavsan, Paul Koning, Chris Trobridge, and Michael Richardson. Also, special thanks to Tim Jenkins, Stephane Beaulieu, and Carson Sutton for many sanity checks along the way.

15. References

[ARCH] Kent, S., Atkinson, R., "Security Architecture for the Internet Protocol", RFC 2401, November 1998

[DOI] Piper, D., "The Internet IP Security Domain of Interpretation for ISAKMP", RFC 2407, November 1998

[EXT-METH] T. Kivinen, "ISAKMP & IKE Extension Methods", draft-ietf-ipsec-ike-ext-meth-03.txt (WORK IN PROGRESS)

[IKE] Harkins, D., Carrel, D., "The Internet Key Exchange (IKE)", RFC 2409, November 1998

[IKEv2] D. Harkins, D. Carrel, "The Internet Key Exchange (IKE)", draft-ietf-ipsec-ike-01.txt (WORK IN PROGRESS)

[IKECFG]R. Pereira, "The ISAKMP Configuration Method", draft-ietf-ipsec-isakmp-cfg-05 (WORK IN PROGRESS)

[ISAKMP]Maughan, D., Schertler, M., Schneider, M., and Turner, J., "Internet Security Association and Key Management Protocol (ISAKMP)", RFC 2408, November 1998

[NOTIFY-DATA] S. Kelly, T. Kivinen, "Content Requirements for ISAKMP Notify Messages", draft-ietf-ipsec-notifymsg-02.txt (WORK IN PROGRESS)

[REVISED-HASH] T. Kivinen, "Fixing IKE Phase 1 Authentication HASH", draft-ietf-ipsec-ike-hash-revised-01.txt (WORK IN PROGRESS)

Appendix A. Future Considerations

One of the goals of the heartbeat protocol was simplicity of design. Although this document is of fairly substantial length, much of the text is of an explanatory nature; the protocol itself is still relatively simple.

For the sake of simplicity, many potentially useful features were omitted from this specification. The most common reason for not including a feature was that it was doubtful whether the feature would actually be used.

Below, we list some of the features which were rejected for this version of the specification. Support for these features may be revisited later, pending comments from implementers.

a) The ability to stop the heartbeats, perhaps by sending a NOTIFY(Stop Heartbeats) message.

This might be useful for the case where a physical link (e.g. dialup connection) goes down gracefully, but we want the ISAKMP SA to remain.

In this case, we would probably also want to have a NOTIFY(Restart Heartbeats) message. This would require special protection against replay attacks.

Currently, the only way to terminate the heartbeats gracefully is to delete the ISAKMP SA.

b) The ability to negotiate an action to take when the heartbeats stop.

Similarly, it might not always be desirable to delete the ISAKMP SA when the heartbeats stop abruptly.

There are a number of possible actions a host might want to take when it ceases to receive proof of liveness. These include: stop billing, hang-up phone, retry later, use alternate route.

c) Support for other types of heartbeat mode (see Appendix B).

In this instance, it seems more important to promote interoperability between vendors than to provide an ultra-flexible protocol.

If it becomes apparent that more than one heartbeat mode is actually needed then new HEARTBEAT_TYPE values will be added. However, preference will be given to solutions that work within the existing heartbeat framework.

d) The use of a query protocol for faster detection of dead peers.

The use of a request/reply protocol for transport of heartbeats is sub-optimal. However, once the peer has been flagged as 'probably dead' (because a heartbeat has been missed), a replay-protected

request/reply protocol could be used to safely speed up the timeout process.

Using this technique, the timeout interval could be reduced to:

TO_I = HB_I + LP_T x PT_W

One way to add this feature to the heartbeat protocol in an unobtrusive manner would be to add a new notify type, NOTIFY(Missed Heartbeat). This would command the sender to retransmit the heartbeat packet (replay protection would be provided by including the sequence number in the notification).

e) Faster packet throughput (especially in the DoS case).

One of the goals of this protocol was to not reduce the security of existing IPsec protocols. Although there is no precise reason why confidentiality of the heartbeat packet is required, encrypting it gives us a level of security equivalent to that provided by other exchange modes.

Currently, the performance impact of using encryption is unclear. Overall throughput will certainly decrease, but resistance to DoS attacks may improve, depending on the precise set of cryptographic algorithms that is being used.

Due to the large number of heartbeat packets that will be available for replay, some kind of anti-clogging mechanism is needed. In this case, the most effective variety of anti-clogging device is the time-variant (or sequence-based) token. This is a value which will be unpredictable to an adversary, but easy to calculate (or predict) for the receiver.

The current heartbeat format implements this feature by including an encrypted copy of the sequence number early in the packet. However, this technique has sub-optimal performance characteristics because a prf must be calculated (to generate the IV) before the spoofed packet can be discarded.

For optimal performance against DoS attacks, the anti-clogging token should be sent as a plaintext value, and the receiver should calculate the expected value ahead of time (or set of possible expected values).

One obvious way to accomplish this would be to generate the message ids sequentially, based on a pre-shared prf algorithm. This is a very fast packet-rejection technique, which could potentially also be applied to [IKE] exchanges such as quick mode (if the notion of a

lost packet tolerance was also added to [IKE]).

Appendix B. Other Dead Peer Detection Techniques

Before settling on the current heartbeat protocol, we explored a number of different general approaches. These are listed below, along with the reasons why they were not chosen.

B.1 Terminology Used in This Section

The following terms are used to describe potential heartbeat protocols:

A unidirectional protocol is one in which packets are sent in only one direction (this implies that the heartbeat exchange must be a one packet exchange).

A request/reply protocol is one in which the sender proves his liveness by responding to a query from the receiver.

A stateful protocol is one in which the sender and receiver maintain a shared heartbeat state.

A stateless protocol is the opposite. Generally only the receiver needs to keep a state.

A phase 1 protocol is one in which the heartbeats are sent under the protection of an ISAKMP SA.

A phase 2 protocol is one in which the heartbeats are sent under the protection of an IPsec SA.

An out of band (OOB) protocol is one in which the heartbeats are authenticated using a public key signature.

An insecure protocol is one in which the heartbeats are not authenticated.

The heartbeat protocol described in this document is of the stateful unidirectional phase 1 variety.

B.2 Design Alternatives

The Insecure Heartbeat (a.k.a. clear ping):

The problem with this heartbeat mode is that it is insecure. It is undesirable for a security protocol to use an insecure heartbeat mechanism.

The Out-of-Band Heartbeat:

This heartbeat mode saves time and state on the sender, but consumes valuable time and state on the receiver. This makes it particularly vulnerable to DoS attacks. Since a session key is not used, key exposure is a concern. Also, identity protection and non-reputability are not provided.

The Phase 2 Heartbeat:

This heartbeat mode has many advantages. Unfortunately, it requires extra complexity to negotiate. If negotiation is not used, the peer system must have policy holes to let the packets through.

This mode allows the host to save memory if the heartbeats are mixed in with user traffic, but this behaviour makes it difficult to maintain billing information such as byte counts. If a dedicated SA is used for heartbeats then this memory advantage is nullified.

The Stateless (Request/Reply) Phase 1 Heartbeat:

This heartbeat mode is simple to implement, but it is very vulnerable to DoS attacks. If the sender does not keep a state, he cannot detect replayed heartbeat requests.

The Stateful Request/Reply Phase 1 Heartbeat:

When used properly (with a sequence number and a query interval), this heartbeat mode is similar to the one described in this document. The main difference is that it requires double the bandwidth to do the same thing.

Authors' Addresses

Andrew Krywaniuk
Alcatel Networks Corporation
600 March Rd.
Kanata, ON
Canada, K2K 2P5
+1 (613) 599-3610 x4237
E-mail: andrew.krywaniuk@alcatel.com

Tero Kivinen
SSH Communications Security Ltd.
Tekniikantie 12
FIN-02150 ESPOO
Finland
E-mail: kivinen@ssh.fi

The IPsec working group can be contacted via the IPsec working group's mailing list (ipsec@lists.tislabs.com) or through its chair:

Theodore Y. Ts'o
tytso@MIT.EDU
Massachusetts Institute of Technology

Expiration

This document expires January 14, 2001.