

IPsec Channels: Connection Latching
draft-ietf-btnc-connection-latching-02.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/1id-abstracts.txt>.

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

This Internet-Draft will expire on March 15, 2008.

Copyright Notice

Copyright (C) The IETF Trust (2007).

Abstract

This document specifies, abstractly, how to interface applications and transport protocols with IPsec so as to create "channels" by "latching" "connections" (packet flows) to certain IPsec Security Association (SA) parameters for the lifetime of the connections. This can be used to protect applications against accidentally exposing live packet flows to unintended peers, whether as the result of a reconfiguration of IPsec or as the result of using weak peer identity to peer address associations.

Weak association of peer ID and peer addresses is at the core of Better Than Nothing Security (BTNS), thus connection latching can add a significant measure of protection to BTNS IPsec nodes. A model of of connection latching based on a modification to the child SA authorization process is given.

Williams

Expires March 15, 2008

[Page 2]

Table of Contents

1.	Introduction	4
1.1.	Conventions used in this document	4
2.	Connection Latching	5
2.1.	Normative Model: ULP interfaces to the key manager and child SA authorization process extensions	7
2.2.	Using Intimate Interfaces Between ULPs and IPsec	10
2.3.	Non-native mode IPsec	11
3.	Optional protection	12
4.	Security Considerations	13
5.	IANA Considerations	14
6.	Acknowledgements	15
7.	References	16
7.1.	Normative References	16
7.2.	Informative References	16
	Author's Address	17
	Intellectual Property and Copyright Statements	18

1. Introduction

IPsec protects packets with little or no regard for stateful packet flows associated with upper layer protocols (ULPs). This exposes applications that rely on IPsec for session protection to risks associated with changing IPsec configurations, configurations that allow multiple peers access to the same addresses, and/or weak association of peer IDs and their addresses. The latter can occur as a result of "wildcard" matching in the IPsec Peer Authorization Database (PAD), particularly when BTNS [[I-D.ietf-btms-prob-and-applic](#)] is used.

A method is desired for creating "IPsec channels," that is, packet flows predictably protected for their duration, even in the face of IPsec reconfiguration or weak association of peer IDs and addresses. The methods outlined below achieve this by interfacing ULPs and applications to IPsec and using these interfaces to bind ("latch") connections to peer IDs and SA parameters.

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. Connection Latching

An "IPsec channel" is a packet flow associated with a ULP control block, such as a TCP connection, where all the packets are protected by IPsec SAs such that:

- o the peer's identity is the same for the lifetime of the packet flow
- o the quality of IPsec protection used for the packet flow's individual packets is the same for all of them for the lifetime of the packet flow

An IPsec channel is created when the associated packet flow is created. This can be the result of a local operation (e.g., a connect()) that causes the initial outgoing packet for that flow to be sent, or it can be the result of receiving the first/initiating packet for that flow (e.g., a TCP SYN packet).

IPsec channels are created by "latching" various parameters listed below to a ULP connection when the connections are created. The REQUIRED set of parameters bound in IPsec channels is:

- o Type of protection: confidentiality and/or integrity protection;
- o Transport mode vs. tunnel mode;
- o Quality of protection: cryptographic algorithm suites, key lengths, and replay protection;
- o Peer identity: peers' asserted and authorized IDs, as per the IPsec processing model [[RFC4301](#)] and BTNS [[I-D.ietf-btns-core](#)].

Implementations SHOULD provide applications with APIs for inquiring whether a connection is latched and what the latched parameters are. Implementations SHOULD provide applications with some control, through application programming interfaces (APIs) [[I-D.ietf-btns-abstract-api](#)], over what quality of protection, or the expected identity of a peer. If an application does not use such interfaces then it will obtain default quality of protection derived from system policy. Implementations MAY create IPsec channels automatically by default when the application does not request an IPsec channel.

IPsec channels have the following states:

- o Listener

- o Larval (in the process of being established)
- o Established
- o Failed

Requirements and recommendations:

- o If an IPsec channel is desired then packets for a given connection MUST NOT be sent until the channel is established.
- o If an IPsec channel is desired then inbound packets for a given connection MUST NOT be accepted (they must be dropped) until the channel is established.
- o Once an IPsec channel is established packets for the latched connection MUST NOT be sent unprotected nor protected by an SA that does not match the latched parameters.
- o Once an IPsec channel is established packets for the latched connection MUST NOT be accepted unprotected nor protected by an SA that does not match the latched parameters (i.e., such packets must be dropped).
- o Native implementations SHOULD provide programming interfaces for inquiring the values of the parameters latched in a connection.
- o Implementations that provide such programming interfaces MUST make available to applications all relevant information about a peer's ID, including authentication information. This includes the peer certificate, when one is used, and the trust anchor that it was validated to.
- o Implementations that provide such programming interfaces MUST make available to applications NAT-related information about the peer: whether it is behind a NAT and, if it is, the inner and outer tunnel addresses of the peer.
- o Native implementations SHOULD provide programming interfaces for setting the values of the parameters to be latched in a connection that will be initiated or accepted, but these interfaces MUST limit what values applications may request according to system policy (i.e., the IPsec PAD and SPD) and the application's privilege.

(Typical system policy may not allow applications any freedom here. Policy extensions allowing for optional protection are described in [Section 3](#).)

Williams

Expires March 15, 2008

[Page 6]

- o The parameters latched in an IPsec channel MUST remain unchanged once the channel is established.
- o Timeouts while establishing an SA with parameters that match a those latched into an IPsec channel MUST be treated as packet loss (as happens, for example, when a network partitions); normal ULP and/or application timeout handling and retransmission considerations apply. Failure to establish an appropriate SA for an IPsec channel MAY be communicated to the ULP and application (e.g., as though the connection had been reset)
- o Implementations that have a restartable key management process (or "daemon") MUST arrange for existing latched connections to either be reset and disconnected, or for them to survive the restart of key exchange processes. (This is implied by the above requirements.)
- o Any IPsec channel created with a given peer while another distinct, established IPsec channel exists with the same source and destination addresses SHOULD be bound to the same peer.

We describe two models (one normative) of IPsec channels for native IPsec implementations. Both models should suffice for all-software native implementations of IPsec. One the other or both models should be workable for most native implementations where part of the IPsec stack is implemented in hardware. The normative model is based on abstract programming interfaces between ULPs and the key management component of IPsec, plus a modification to the child SA authorization process. The second model is based on abstract programming interfaces between ULPs and the ESP/AH layer in the IP stack. Both models imply extensions to any PF_KEY-like protocols [[RFC2367](#)] that may be used internally by the implementation.

We also provide a model for non-native implementations, such as bump-in-the-stack (BITS) and SG implementations. The connection latching model for non-native implementations is not full-featured as it depends on estimating packet flow state, which may not always be possible. Nor can non-native IPsec implementations be expected to provide APIs related to connection latching. As such this third model is not suitable for channel binding applications [[I-D.williams-on-channel-binding](#)].

[2.1](#). Normative Model: ULP interfaces to the key manager and child SA authorization process extensions

This section is NORMATIVE.

In this section we describe connection latching in terms of an

interface between ULPs and the key manager component of a native IPsec implementation. Abstract interfaces for creating, inquiring about, and releasing IPsec channels are described.

This model requires an extension to the IPsec child SA authorization process such that no SAs conflicting with a connection latch are allowed. Normally the IPsec processing model does allow SAs with different peers but overlapping traffic selectors -- behaviour that, in this model, directly violates the requirements for connection latching.

The ULP interfaces to the IPsec PAD database are as follows:

- o Create a connection latch listener object for a ULP 3-tuple (local address, protocol and local port number). This operation succeeds whenever there are no other connection latch listeners for the same 3-tuple. Connection latch listener objects can result in connection latch objects when a child SA is created whose traffic selectors encompass the given 3-tuple.
- o Create a connection latch object for a ULP 5-tuple (local and remote address, protocol and local and remote port numbers). This operation succeeds when no conflicting connection latch objects exist and when there exist no child SAs encompassing the given 5-tuple or when all such SAs are with the same peer and equal quality of protection.
- o Destroy a connection latch listener object.
- o Destroy a connection latch object.
- o Inquire whether a connection latch exists for a given 5-tuple, its state, and its latched parameters.

The IPsec processing model is modified as follows:

1. The API described above is a new service of the IPsec key manager. In particular the IPsec key manager has to reject new connection latches that conflict with others or with current SAD state.
2. At child SA creation time the IPsec key manager MUST reject child SA proposals that would conflict with an established connection latch, or else it MUST narrow such proposed child SAs such that the resulting SAs do not conflict with established connection latches.

Implementations SHOULD provide a flag for PAD entries such that PAD

entries so flagged will result in the key manager rejecting any connection latch requests with remote addresses which peers matching those PAD entries may assert. This flag makes it possible to preserve traditional IPsec child SA authorization semantics where desired. Alternatively implementations MAY provide a flag to disable or enable connection latching altogether.

ULPs create latched connections by interfacing with IPsec below as follows:

- o For listening end-points the ULP will request a connection latch listener object for the ULP listener's 3-tuple. Any latching parameters requested by the application should be passed along.
- o When initiating a connection the ULP will request a connection latch object for the connection's 5-tuple. Any latching parameters requested by the application should be passed along.
- o When a latched connection is torn down and no further packets are expected for it then the ULP will request that the connection latch object be destroyed.
- o When tearing down a listener the ULP will request that the connection latch listener object be destroyed.
- o When a ULP listener rejects connections for non-existent connections the ULP will request the destruction of any connection latch objects that may have been created as a result of the peer's attempt to open the connection.

The main benefit of this model of connection latching is that it accommodates IPsec implementations where ESP/AH handling is implemented in hardware (for all or a subset of the host's SAD), but where the hardware does not support tagging inbound packets with the SAD entries corresponding to the SAs that protected them.

Note that there is a race condition in this method of connection latching: packets may race with the ULP and the IPsec key manager's manipulation of connection latch objects and SAD entries. As a result ULPs may not be able to trust some packets even though a suitable connection latch object may exist. Implementations MUST prevent such races. One method to prevent these races is to tag packets passed up by the ESP/AH layer with a key manager state version number that is monotonically incremented every time that connection latching state changes; this version number must be incremented atomically relative to the SAD, including SAD subsets stored on IPsec offload hardware. Other methods may be possible, including dropping packets that arrive within a certain amount of

time since the creation/destruction of connection latch objects (e.g., if the maximum latency within the key manager and IP stack is known and guaranteed).

2.2. Using Intimate Interfaces Between ULPs and IPsec

In this section we describe connection latching in terms of interfaces between ULPs and IPsec based on tagging packets as they go up and down the IP stack.

This section is INFORMATIVE.

The ULPs and IPsec interface through a local packet tagging scheme (the tags don't appear on the wire):

- o The IPsec layer tags all inbound protected packets addressed to the host with the index of the SAD entry corresponding to the SA that protected the packet.
- o The IPsec layer understands two types of tags on outbound packets:
 - * a tag specifying a set of latched parameters (peer ID, quality of protection, etc...) that the IPsec layer will use to find or acquire an appropriate SA for protecting the outbound packet (else IPsec will drop the packet;
 - * a tag requesting feedback about the SA used to protect the outgoing packet, if any.

ULPs create latched connections by interfacing with IPsec below as follows:

- o When the ULP passes a connection's initiating packet to IP the ULP requests feedback about the SA used to protect the outgoing packet, if any, and may specify latching parameters requested by the application. If the packet is protected by IPsec then the ULP records certain parameters of the SA used to protect it in the connection's transmission control block (TCB).
- o When a ULP receives a connection's initiating packet it processes the IPsec tag of the packet, and it records in the connection's TCB the parameters of the SA that should be latched.

Once SA parameters are recorded in a connection's TCB the ULP enforces the connection's latch, or binding, to these parameters as follows:

- o The ULP processes the IPsec tag of all inbound packets for a given

Williams

Expires March 15, 2008

[Page 10]

connection and checks that the SAs used to protect input packets match the connection latches recorded in the TCBs; packets which are not so protected are dropped.

- o The ULP always requests that outgoing packets be protected by SAs that match the latched connection by appropriately tagging outbound packets.

The main benefit of this model of connection latching is its simplicity. For example, no changes need be made to the child SA authorization process. However, this model of connection latching is not workable with ESP/AH offload hardware that does not support the packet tagging scheme described above.

2.3. Non-native mode IPsec

[Fill this out. Basically, for BITS/BITW/SG implementations connection latching requires inspecting packets to discern ULP connection state, recording such state, and establishing associated connection latches. Like all stateful middle-boxes this suffers from the inability of the middle-box to interact with the applications. For example, connection death may be difficult to ascertain. Nor can channel binding applications work with channels maintained by proxy without being able to communicate (securely) about it with the proxy.]

[Sam requested this section offline, and believe we need a PAD entry flag to indicate which PAD entries' addresses (child SA constraints) are subject to connection latching, and which are not. Sam believes this is needed in the native IPsec model based on extending the child SA authorization process; I disagree. -Nico]

3. Optional protection

Given IPsec APIs an application could request that a connection's packets be protected where they would otherwise be bypassed; that is, applications could override BYPASS policy. Locally privileged applications could request that their connections' packets be bypassed rather than protected; that is, privileged applications could override PROTECT policy. We call this "optional protection." Note that optional protection as described here does not provide a way to override DISCARD policy.

Both native IPsec models of connection latching can be extended to support optional protection. With the model described in [Section 2.2](#) optional protection comes naturally: the IPsec layer need only check that the protection requested for outbound packets meets or exceeds the quality of protection, if any, required by the SPD. Similarly, for the model described in [Section 2.1](#) the check that requested protection meets or exceeds that required by the SPD is performed by the IPsec key manager when creating connection latch and connection latch listener objects.

4. Security Considerations

Connection latching protects only individual connections from weak peer ID->address binding, IPsec configuration changes, and from configurations that allow multiple peers to assert the same addresses, but it does not ensure that any two connections with the same end-point addresses, even if one established while the other is alive, will have the same latched peer IDs. In other words, applications that use multiple concurrent connections between two given nodes are not protected any more or less by use of IPsec connection latching than by use of IPsec alone. Such multi-connection applications can, however, examine the latched SA parameters of each connection to ensure that each every connection with the same end-point addresses also has the same end-point IPsec IDs.

IPsec channels are a pre-requisite for channel binding [[I-D.williams-on-channel-binding](#)] to IPsec. Connection latching provides such channels, but the process of binding IPsec channels (latched connections) to authentication at application layers is not specified herein.

Without IPsec APIs connection latching provides marginal security benefits over traditional IPsec. Such APIs are not described herein; see [[I-D.ietf-btms-abstract-api](#)].

5. IANA Considerations

There are not IANA considerations for this document.

6. Acknowledgements

The author thanks Michael Richardson for all his help, as well as Stephen Kent, Sam Hartman, Bill Sommerfeld, Dan McDonald, and many others who've participated in the BTNS WG or who've answered questions about IPsec, connection latching implementations, etc...

7. References

7.1. Normative References

- [I-D.ietf-btms-abstract-api]
Richardson, M., "An interface between applications and keying systems", [draft-ietf-btms-abstract-api-00](#) (work in progress), June 2007.
- [I-D.ietf-btms-core]
Williams, N. and M. Richardson, "Better-Than-Nothing-Security: An Unauthenticated Mode of IPsec", [draft-ietf-btms-core-05](#) (work in progress), September 2007.
- [I-D.ietf-btms-prob-and-applic]
Touch, J., "Problem and Applicability Statement for Better Than Nothing Security (BTNS)", [draft-ietf-btms-prob-and-applic-05](#) (work in progress), February 2007.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.
- [RFC2367] McDonald, D., Metz, C., and B. Phan, "PF_KEY Key Management API, Version 2", [RFC 2367](#), July 1998.
- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", [RFC 4301](#), December 2005.
- [RFC4306] Kaufman, C., "Internet Key Exchange (IKEv2) Protocol", [RFC 4306](#), December 2005.
- [RFC4322] Richardson, M. and D. Redelmeier, "Opportunistic Encryption using the Internet Key Exchange (IKE)", [RFC 4322](#), December 2005.

7.2. Informative References

- [I-D.bellovin-useipsec]
Bellovin, S., "Guidelines for Mandating the Use of IPsec", [draft-bellovin-useipsec-06](#) (work in progress), February 2007.
- [I-D.williams-on-channel-binding]
Williams, N., "On the Use of Channel Bindings to Secure Channels", [draft-williams-on-channel-binding-04](#) (work in progress), August 2007.

Author's Address

Nicolas Williams
Sun Microsystems
5300 Riata Trace Ct
Austin, TX 78727
US

Email: Nicolas.Williams@sun.com

Full Copyright Statement

Copyright (C) The IETF Trust (2007).

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.

Acknowledgment

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

