NETWORK WORKING GROUP
Internet-Draft
Expires: March 20, 2008

N. Williams
Sun

September 17, 2007

IPsec Channels: Connection Latching
draft-ietf-btns-connection-latching-03.txt

Status of this Memo

Copyright Notice

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.

Table of Contents

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-btns-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:

- the peer's identity is the same for the lifetime of the packet flow
- 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:

- Type of protection: confidentiality and/or integrity protection;
- Transport mode vs. tunnel mode;
- Quality of protection: cryptographic algorithm suites, key lengths, and replay protection;
- 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:

- Listener

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

Requirements and recommendations:

- If an IPsec channel is desired then packets for a given connection MUST NOT be sent until the channel is established.
- 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.
- 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.
- 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).
- Native implementations SHOULD provide programming interfaces for inquiring the values of the parameters latched in a connection.
- 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.
- 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.
- 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.)

- The parameters latched in an IPsec channel MUST remain unchanged once the channel is established.

- 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)

- 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.) IPsec state related to connection latches MUST be torn down when latched connections are torn down, even when the latter is implied, such as at crash/halt/reboot time.

- 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 IPsec (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 adds a service to the IPsec key manager: management of connection latches.

The traditional IPsec processing model allows the concurrent existence of SAs with different peers but overlapping traffic selectors. Such behaviour, in this model, directly violates the requirements for connection latching. We address this problem by requiring that either such conflicts be avoided or that connection latches be broken (and holders informed) when such conflicts arise.

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

- 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.

- 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. The key manager SHOULD attempt to create a suitable SA pair if one does not already exist; if it does then it MUST use the 5-tuple as the initial traffic selectors of the proposed child SAs.

- Destroy a connection latch listener object.

- Destroy a connection latch object.

- Inquire whether a connection latch exists for a given 5-tuple, its state, and its latched parameters.

The API described above is a new service of the IPsec key manager. In particular the IPsec key manager MUST prevent conflicts between latches and SAs as follows:

- connection latches MUST NOT be created if there exist conflicting

SAs in the SAD at the time the connection latch is requested or would be created (from a listener latch);

- child SA proposals that would conflict with an extant connection latch and whose traffic selectors can be narrowed to avoid the conflict MUST be narrowed (see section 2.9 of [RFC4306]);

- where child SA proposals that would conflict with an extant connection latch cannot be narrowed to avoid the conflict the key manager MUST either reject such proposals or it MUST break the connection latch and inform the holder (the ULP) prior to accepting the conflicting SAs.

Additionally, the key manager MUST protect latched connections against SPD changes that would change the quality of protection afforded to a latched connection's traffic, or which would bypass it. When such a configuration change takes place the key manager MUST eithe preserve a logical SPD entry such that the latched connection continues to obtain protection, or the key manager MUST inform the latch holder (ULP) that the latch is no longer in force before the change takes place. To do this the key manager can logically update the SPD as if a PROTECT entry had been added at the head of the SPD-S with traffic selectors matching only the latched connection's 5-tuple, and with processing information taken from the actual SPD entry matched by the connection (possibly augmented by the application's request for additional protection). Such updates of the SPD MUST NOT survive system crashes or reboots.

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

- 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.

- When the ULP receives a packet initiating a connection for a 5-tuple matching a 3-tuple listener latch, then the ULP will ask the key manager whether a 5-tuple connection latch was created. If not then the ULP will either reject the new connection or accept it and inform the application that the new connection is not latched (that it does not represent an IPsec channel).

- 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. If no latch can be created then the ULP will either return an error to the application or continue with the new connection and inform the application that the new connection is not latched.

- 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.

- When tearing down a listener the ULP will request that the connection latch listener object be destroyed.

- When a ULP listener rejects 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.

- When the key manager informs a ULP that a connection latch is no longer valid then the ULP will reset or otherwise terminate the connection and inform the application.

Note that initiating a connection should result in creation of a suitable SA pair in the same manner as described in Section 2.1.

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 indexes of 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):

- 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.

- 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:

- 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).

- 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:

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

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

The receipt of a packet matching a latched connection's 5-tuple, but protected by an SA with an inappropriate peer, should be taken as an indication that the original peer is no longer at the original address and that the connection should be reset, the application

informed, and the connection latch removed.

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 non-native (BITS/BITW/SG, though BITW probably need not apply) implementations, connection latching requires inspecting packets to discern ULP connection state, recording and tracking such state. 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]

2.4. Conflict Resolution

Consider a system, say, an IMAP server, with an IPsec policy allowing all peers with certificates issued by some CA to claim any dynamically allocated address in a local network.

In such an environment a peer might appear using some address, then disappear (e.g., a laptop whose battery runs out) and another peer might later (after the first peer's DHCP lease expires) appear using the same IP address as the first peer. The first peer might have had a long-lived TCP connection open with the server. The new peer might try to open a connection with the same server and with the same 5-tuple as the first peer. The new peer's TCP SYN packet will fail to match the existing connection's latch.

In such cases implementations based on Section 2.1 and Section 2.3 will be unable to narrow the new peer's child SA proposals to avoid a conflict, and must either reject them or terminate the existing connection. Implementations based on Section 2.2 must either drop the new peer's TCP SYN packet, respond with a TCP RST packet, or terminate the existing connection.

Implementors MUST provide termination of the existing connection as the default behaviour in such cases. Implementors MAY provide a configuration option for selecting the other behaviours.

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."

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.

When an application requests, and IPsec permits, either additional protection, or bypassing protection, then the SPD MUST be logically updated such that there exists a suitable SPD entry protecting or bypassing the exact 5-tuple recorded by the corresponding connection latch. Note connection latching must be used to represent bypassed connections too where such connection override system policy. Such updates of the SPD MUST NOT survive system crashes or reboots. See Section 2.1.

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 connection latching does not ensure that any two connections with the same end-point addresses 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 all concurrent connections with the same end-point addresses also have 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-btns-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-btns-abstract-api]
Richardson, M., "An interface between applications and keying systems", draft-ietf-btns-abstract-api-00 (work in progress), June 2007.

[I-D.ietf-btns-core]
Williams, N. and M. Richardson, "Better-Than-Nothing-Security: An Unauthenticated Mode of IPsec", draft-ietf-btns-core-05 (work in progress), September 2007.

[I-D.ietf-btns-prob-and-applic]
Touch, J., "Problem and Applicability Statement for Better Than Nothing Security (BTNS)", draft-ietf-btns-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

Intellectual Property

Acknowledgment

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