Network Working Group
INTERNET-DRAFT

Nicolas Williams
Sun Microsystems
November 2004

# On the Use of Channel Bindings to Secure Channels
<draft-ietf-nfsv4-channel-bindings-01.txt>

## Status of this Memo



## Copyright Notice



## Abstract

This document defines and formalizes the concept of channel bindings to secure layers and defines the actual contents of channel bindings for several secure channels.

The concept of channel bindings allows applications to prove that the end-points of two secure channels are the same by binding authentication at one network layer to the session protection negotiation at a lower network layer. The use of channel bindings allows applications to delegate session protection to lower layers.

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].

## 1. Introduction

[NOTE: This I-D text has been split out from the "The Channel Conjunction Mechanism (CCM) for GSS" I-D, which will be updated soon to define only the CCM-BIND and CCM-MIC GSS-API pseudo-mechanisms and describe their use. CCM-BIND is particularly relevant to the use of channel bindings with GSS-API applications. See draft-ietf-nfsv4-ccm-01.txt.]

Over the years several attempts have been made to delegate session protection at one network layer to another, for performance and/or scalability as well as for design elegance and also to avoid having to reinvent the wheel for every new application or security layer.

The critical security problem to solve in order to achieve such delegation of session protection is always the same: how to ensure that there is no man-in-the-middle (MITM), from the point of view the application, at the lower network layer to which session protection is to be delegated.

Alternative statement of the problem: how does one ensure that the end-points of two secure channels at different network layers are the same?

And there may well be a MITM, particularly if the lower network layer either provides no authentication or if there is no connection between the authentication or principals used at the application and those used at the lower network layer.

Such MITM attacks can be effected by, for example, spoofing IP address lookups (which is possible, for example, when using DNS but not DNSSEC) in a way that the application may not detect but which directs the client application or network stack to connect to a different host than had been intended (e.g., to the MITM's host). Even if such MITM attacks seem particularly difficult to effect, the problem must be solved.

For example: a user decides to use TELNET, with Kerberos V authentication, over TLS to connect to some server but an attacker spoofs the name service lookup and causes the TELNET client to be redirected to some other host which TLS authenticates correctly and where the attacker forwards the connection, with or without TLS, to the server that the user had intended. In this example there is an

MITM from the point of view of the application (TELNET), even though there is no MITM as far as TLS is concerned. The TELNET client and

server cannot assume that there is no MITM and so cannot leverage the protection afforded by the TLS channel, unless they prove to each other that there is no MITM.

A solution to this problem is highly desirable, particularly where multi-user applications are run over secure network layers (e.g., NFS over IPsec). For such applications the authentication model used at the application layer (usually user<->server) is generally very different from that used by secure, lower network layers, such as IPsec (usually client<->server or single-user<->server), and may even use different authentication infrastructures altogether (e.g., Kerberos V for the application layer, x.509 certificates at the lower layer). Such applications cannot generally leverage the security provided by the lower network layers, which, if they could, would allow them to offload session security to the secure lower layer.

One solution involves ensuring the use of secure name services for hostname to network address translation and the use of secure networks (e.g., IPsec). This approach can prevent the MITM attack described above, but does not offer applications any guarantees that there is no MITM in the lower layer.

Another solution is to use "channel bindings" (a GSS-API concept [RFC2743]) to bind authentication at application layers to secure transports at lower layers in the network stack. This solution is only applicable to applications that provide for user authentication.

"Channel bindings" are data which securely identify a secure channel such that, when verified to match on both endpoints of end-to-end application connections, leave no doubt that the endpoints of two secure channels (the one identified by the bindings and the one used to exchange/verify the bindings) are the same.

Because many applications exist which provide for authentication at the application layer, because many such applications use generic authentication frameworks, such as the GSS-API and SASL and are already deployed along with a common authentication infrastructure (e.g., Kerberos V, PKI, etc...), because such applications exist which multiplex multiple users onto a single session (and so cannot leverage network [e.g., IKE] authentication), the use of channel bindings is an elegant solution even where secure name services and networks are deployed.

A formal definition of the channel bindings concept is given below, as well as the specific formulation of channel bindings for various protocols that provide for session security.

## 2. Definitions

The GSS-API [RFC2743] is a generic interface to GSS-API security mechanisms which provides for authentication and session cryptographic protection. One facility provided by the GSS-API is a concept of "channel bindings" which consists of some data which must

be provided, if at all, by initiators and acceptors and which the GSS-API security mechanisms ensure are the same for both, the initiator and acceptor of any given GSS-API security context - if the channel bindings provided by them do not match then the mechanism fails to establish a security context.

- Channel bindings

  Generally some data which names a channel or its end-points.

- Channel bindings to secure channels

  Channel bindings that securely identify a secure channel or its end-points.

  Applications can exchange authenticated, integrity-protected verifiers of their same channel bindings data to prove that the end-points of the channel identified by the channel bindings are the same as the application endpoints and thus, there can be no MITM at the lower layer.

  More formally, there are two types of channel bindings:

  - bindings that name a channel in a cryptographically secure manner (e.g., the session ID in SSHv2; see below)

  - bindings that name the authenticated end-points of a channel (e.g., as in IPsec; see below)

    Bindings that name a channel

    - MUST be cryptographically bound to the key exchange of the secure session

    - MUST be cryptographically bound to all potentially un-authenticated plaintext used for negotiation of the secure session (e.g., algorithm negotiations)

  and

  - users of channel bindings MUST exchange authenticated, integrity protected channel bindings data or signatures thereof (such exchanges MAY also be confidentiality protected)

  Additionally, the channel represented by the bindings MUST provide a cryptographically secure key exchange (and re-keying) and channel setup negotiation, and it MUST provide at least cryptographically secure data integrity protection services.

Channel bindings data SHOULD NOT be constructed in such a way that their exchange requires confidentiality protection.

No channel bindings described herein require confidentiality protected exchanges.

The security of channel bindings depends on the security of:

- the authentication and integrity protection technology used to protect the channel bindings exchanges at the application layers

- the security of the channels identified by the channel bindings

- the security of the channel bindings construction

o Channel bindings to network addresses

The GSS-API originally defined only channel bindings to network addresses. Such channel bindings, of course, are generally not cryptographically secure.

For channel bindings to network addresses to be secure the application peers MUST be able to verify and ensure that network communications between them are secured and that there is no MITM - which generally means that the application peers MUST be able to interpret and authorize identities authenticated by the network and MUST be able to protect the methods by which they obtain the network addresses in the first place.

In practice channel bindings to network addresses have mostly just caused trouble with Network Address Translation (NAT).

## 3. Authentication protocols and channel bindings

Some authentication services provide for channel bindings, such as the GSS-API and some GSS-API mechanisms - others do not, such as SASL. Where suitable channel bindings facilities are not provided application protocol designers may include a separate, protected (where the authentication service provides message protection services) exchange of channel bindings material

### 3.1. The GSS-API and channel bindings

The GSS-API provides for the use of channel bindings during initialization of GSS-API security contexts, though GSS-API mechanisms are not required to support this facility.

This channel bindings facility is defined in detail in RFC2744.

Unfortunately, the use of GSS-API channel bindings is generally not

negotiated by GSS-API mechanisms, therefore GSS-API applications must agree a priori on the use of channel bindings or otherwise negotiate the use of channel bindings.

Fortunately, it is possible to design GSS-API pseudo-mechanisms that simply wrap around existing mechanisms for the purpose of allowing applications to negotiate the use of channel bindings within their existing methods for negotiating GSS-API mechanisms. For example, NFSv4 [RFC3530] provides its own GSS-API mechanism negotiation, as does the MOUNT protocol for NFSv2/3 [RFC....]. [NOTE: This is an indirect reference to the Channel Conjunction Mechanism (CCM).]

### 3.2. SASL and channel bindings

SASL does not provide for the use of channel bindings during initialization of SASL contexts.

SASL applications MAY define their own exchange of integrity-protected channel bindings using established SASL integrity layers.

Alternatively, SASL applications MAY use the GSS-* SASL mechanisms (which correspond to GSS-API mechanisms) to ensure the use of channel bindings through the GSS-API's facilities.

### 3.3. Kerberos V and channel bindings

Kerberos V does not provide for use of channel bindings, thus the same general approach given above (post-authentication protected channel bindings exchange) applies to Kerberos V as well.

However, Kerberos V AP client applications also MAY use the AP-REQ's Authenticator's "checksum" field to send a hash of channel bindings material to Kerberos V AP servers. Unfortunately, there is no slot in the AP-REP message for carrying the AP server's channel bindings (which justifies the statement that Kerberos V does not provide a channel bindings facility), so Kerberos V applications MUST establish a convention with regards to AP servers' handling of AP-REQ checksum data - and such applications have to trust the servers to respond with suitable error messages to AP-REQs bearing incorrect channel bindings.

## 4. Channel bindings to secure layers

Not every secure session protocol or interface provides for secure channels, and not every secure session protocol provides data suitable for use as channel bindings.

### 4.1. Bindings to SSHv2 channels

SSHv2 provides both, a secure channel and material (the SSHv2 "session ID") that is suitable for use as channel bindings.

Thus it is RECOMMENDED that the SSHv2 "session ID" be used as the

channel bindings for SSHv2.

## 4.2. Bindings to TLS channels

TLS provides both, a secure channel and material (the TLS "finished" messages), that is suitable for use as channel bindings.

Thus it is RECOMMENDED that the concatenation of the client's and server's "finished" messages, in that order, be used as the channel bindings for TLS.

Note that the TLS "session ID," in spite of being named similarly to the SSHv2 session ID, is not suitable for use as channel bindings because it is assigned by the server, so a MITM could assign the same session ID on the client side as it gets from the server.

### 4.3. Bindings to IPsec

IPsec does not provide a way to reliably name a channel regardless of what key exchange protocol is used.

Therefore, the only reliable way to construct channel bindings to IPsec is to use the identities authenticated by the IPsec key exchange protocol for the given channel.

New interfaces [IPSP-APIREQ] are required by which applications can create IPsec "channels."

The basic idea is to use the interfaces described in [IPSP-APIREQ] to dynamically alter the SPD to reflect the requested bindings for the requested connections and then use the authenticated IDs as the identity of the channel.

This approach does not name the channel directly, but no MITM can ensure that the authenticated IDs used as channel bindings match on both end-points unless the MITM has stolen or broken the IPsec credentials and/or authentication protocol. This is sufficient.

#### 4.3.1. Interfaces for creating IPsec channels

In order to build an IPsec channel some additional application programming interfaces are needed to:

- indicate that an as yet unconnected channel is to be bound to IPsec IDs and
  - explicitly specify one, the other or both of those IDs
  - implicitly specify one, the other or both of those IDs (e.g., the ID corresponding to the current application program instance)
  - indirectly specify one, the other or both of those IDs (e.g., a hostname)

and/or

- discover the IPsec IDs to which a channel is bound

For connection-less datagram transports the IDs to be used need to be specified/discovered on a per-datagram basis.

See [IPSP-APIREQ].

### 4.4. Bindings to other types of channels

For secure session protocols that do not provide material suitable for use as channel bindings such material SHOULD be constructed by concatenating the octets from the messages exchanged during the initialization of a session in the chronological order in which they were exchanged and processed (which requires synchronous session initialization), or a strong hash thereof (such as SHA-1).

Some secure session protocols do not provide a secure channel but which do provide per-message integrity or confidentiality protection services. It is up to the network layers that use such protocols to build channels from such services; applications MUST NOT delegate session cryptographic protection to network layers that do not provide a secure channel.

Kerberos V, certain GSS-API and SASL mechanisms, all provide session cryptographic protection and the necessary key exchange, but they provide neither a channel nor material suitable for use as channel bindings.

Thus the RECOMMENDED channel bindings for channels protected by Kerberos V consist of a SHA-1 hash of the concatenated octets of the AP-REQ and AP-REP messages, in that order (or, for user-to-user exchanges, the various messages exchanged, including the ticket request, ticket and AP messages, in the order in which they were generated and processed) used to initialize the channel's cryptographic protection.

Similarly for channels protected by GSS-API security contexts the RECOMMENDED channel bindings consist of a SHA-1 hash of the concatenated octets of the context tokens exchanged to setup a GSS-API security context in the order in which they were generated and processed (i.e., starting with the initiator's initial context token followed by the acceptor's reply token, if any, followed by the initiator's reply token, if any, etc...).

## 5. Benefits of channel bindings to secure channels

The use of channel bindings to delegate session cryptographic protection include:

- Performance improvements by avoiding double protection in cases where IPsec is in use and applications provide their own secure

channels.

- Performance improvements by leveraging hardware-accelerated IPsec.

- Performance improvements by allowing RDDP hardware offloading to be integrated with IPsec hardware acceleration. If protocols layered above RDDP use privacy protection then RDDP offload cannot be done, thus by using channel bindings to IPsec the privacy protection is moved to IPsec, which is layered below RDDP, so RDDP can address application protocol data that's in cleartext relative to the RDDP headers.

- Latency improvements for applications that multiplex multiple users onto a single channel, such as NFS w/ RPCSEC_GSS.

## 6. Security considerations

When delegating session protection from one layer to another, one will almost certainly be making some session security trade-offs, such as using weaker data encryption/authentication modes. Implementors and administrators SHOULD understand these trade-offs.

Channel bindings cannot and MUST NOT be used without mutual authentication (of client/user/initiator and server/user/acceptor) and/or without integrity-protected, authenticated exchange of channel bindings material.

Anonymous secure channels SHOULD NOT be used without authentication and corresponding use of channel bindings (to the anonymous secure channels) at higher network layers, or for any purposes other than opportunistic encryption, since such channels provide no authenticated protection on their own.

The security of channel bindings depends on the security of the channels, the construction of the bindings and the security of the authentication and integrity protection used to exchange channel bindings.

## 7. References

### 7.1. Informative references

[Needs references to NFSv2/3 use of RPCSEC_GSS, to NFSv4, to SCTP, and, possibly, to DNS, DNSSEC, TELNET, SPNEGO, SSHv2 gss keyex, and CCM.]

[TELNET]
J. Postel, J.K. Reynolds, RFC0854 (STD0008): "Telnet Protocol Specification," May 1993, Status: Standard.

[DNS]
P.V. Mockapetris, RFC1035 (STD0013): "Domain names - implementation and specification," November 1987, Status:

Standard.

[DNSSEC]

B. Wellington, RFC3008: "Domain Name System Security (DNSSEC) Signing Authority," November 2000, Status: Proposed Standard.

[RFC2203]
M. Eisler, A. Chiu, L. Ling, RFC2203: "RPCSEC_GSS Protocol Specification," September 1997, Status: Proposed Standard.

[RFC2623]
M. Eisler, "NFS Version 2 and Version 3 Security Issues and the NFS Protocol's Use of RPCSEC_GSS and Kerberos V5," June 1999, Status: Proposed Standard.

[NFSv4]
S. Shepler, et. al., RFC3530: "Network File System (NFS) version 4 Protocol," April 2003, Status: Proposed Standard.

[SPNEGO]
E. Baize, D. Pinkas, RFC2478: "The Simple and Protected GSS-API Negotiation Mechanism," December 1998, Status: Proposed Standard.

[CCM]
M. Eisler, N. Williams, Internet-Draft: "The Channel Conjunction Mechanism (CCM) for GSS," May 2003, Status: Internet-Draft.

...

## 7.2. Normative references

[Needs references to RFC2119, RFC2026, the GSS-API (RFCs 2743 & 2744), SASL, SSHv2, IKEv2, IPsec, Kerberos V, ...]

[RFC2026]
S. Bradner, RFC2026: "The Internet Standard Process - Revision 3," October 1996, Obsoletes - RFC 1602, Status: Best Current Practice.

[RFC2119]
S. Bradner, RFC2119 (BCP14): "Key words for use in RFCs to Indicate Requirement Levels," March 1997, Status: Best Current Practice.

[RFC2743]
J. Linn, RFC2743: "Generic Security Service Application Program Interface Version 2, Update 1," January 2000, Status: Proposed Standard.

[RFC2744]
J. Wray, RFC2744: "Generic Security Service API Version 2 : C-bindings," January 2000, Status: Proposed Standard.

[IPSP-APIREQ]
W. Sommerfeld, draft-ietf-ipsp-ipsec-apireq-00: "Requirements for an IPsec API," June 2003, Status: Draft.

...

## 8. Acknowledgements

The author would like to thank Mike Eisler for his work on the Channel Conjunction Mechanism I-D and for bringing the problem to a head, Sam Hartman for pointing out that channel bindings provide a general solution to the channel binding problem, Jeff Altman for his suggestion of using the TLS finished messages as the TLS channel bindings, as well as Bill Sommerfeld, Radia Perlman for their most helpful comments.

## 9. Author's Address

Nicolas Williams
Sun Microsystems
5300 Riata Trace Ct
Austin, TX 78727
Email: nicolas.williams@sun.com

Full Copyright Statement



Acknowledgement

Funding for the RFC Editor function is currently provided by the
Internet Society.