Network Working Group
Request for Comments: 5056
Category: Standards Track

N. Williams
Sun

November 2007

On the Use of Channel Bindings to Secure Channels

Status of This Memo

Abstract

The concept of channel binding allows applications to establish that the two end-points of a secure channel at one network layer are the same as at a higher layer by binding authentication at the higher layer to the channel at the lower layer. This allows applications to delegate session protection to lower layers, which has various performance benefits.

This document discusses and formalizes the concept of channel binding to secure channels.

Table of Contents

1. Introduction

In a number of situations, it is useful for an application to be able to handle authentication within the application layer, while simultaneously being able to utilize session or transport security at a lower network layer. For example, IPsec [RFC4301] [RFC4303] [RFC4302] is amenable to being accelerated in hardware to handle very high link speeds, but IPsec key exchange protocols and the IPsec architecture are not as amenable to use as a security mechanism within applications, particularly applications that have users as clients. A method of combining security at both layers is therefore attractive. To enable this to be done securely, it is necessary to "bind" the mechanisms together -- so as to avoid man-in-the-middle vulnerabilities and enable the mechanisms to be integrated in a seamless way. This is the objective of "Channel Bindings".

The term "channel binding", as used in this document, derives from the Generic Security Service Application Program Interface (GSS-API) [RFC2743], which has a channel binding facility that was intended for binding GSS-API authentication to secure channels at lower network layers. The purpose and benefits of the GSS-API channel binding facility were not discussed at length, and some details were left unspecified. Now we find that this concept can be very useful, therefore we begin with a generalization and formalization of "channel binding" independent of the GSS-API.

Although inspired by and derived from the GSS-API, the notion of channel binding described herein is not at all limited to use by GSS-API applications. We envision use of channel binding by applications that utilize other security frameworks, such as Simple Authentication and Security Layer (SASL) [RFC4422] and even protocols that provide their own authentication mechanisms (e.g., the Key Distribution Center (KDC) exchanges of Kerberos V [RFC4120]). We also envision use of the notion of channel binding in the analysis of security protocols.

The main goal of channel binding is to be able to delegate cryptographic session protection to network layers below the application in hopes of being able to better leverage hardware implementations of cryptographic protocols. Section 5 describes some intended uses of channel binding. Also, some applications may benefit by reducing the amount of active cryptographic state, thus reducing overhead in accessing such state and, therefore, the impact of security on latency.

The critical security problem to solve in order to achieve such delegation of session protection is ensuring 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.

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

Even if such MITM attacks seem particularly difficult to effect, the attacks must be prevented for certain applications to be able to make effective use of technologies such as IPsec [RFC2401] [RFC4301] or HTTP with TLS [RFC4346] in certain contexts (e.g., when there is no authentication to speak of, or when one node's set of trust anchors is too weak to believe that it can authenticate its peers). Additionally, secure channels that are susceptible to MITM attacks because they provide no useful end-point authentication are useful when combined with application-layer authentication (otherwise they are only somewhat "better than nothing" -- see Better Than Nothing Security (BTNS) [BTNS-AS]).

For example, Internet Small Computer Systems Interface (iSCSI) [RFC3720] provides for application-layer authentication (e.g., using Kerberos V), but relies on IPsec for transport protection; iSCSI does not provide a binding between the two. iSCSI initiators have to be careful to make sure that the name of the server authenticated at the application layer and the name of the peer at the IPsec layer match -- an informal form of channel binding.

This document describes a solution: the use of "channel binding" to bind authentication at application layers to secure sessions at lower layers in the network stack.

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. Definitions

- Secure channel: a packet, datagram, octet stream connection, or sequence of connections between two end-points that affords cryptographic integrity and, optionally, confidentiality to data exchanged over it. We assume that the channel is secure -- if an attacker can successfully cryptanalyze a channel's session keys, for example, then the channel is not secure.

- Channel binding: the process of establishing that no man-in-the-middle exists between two end-points that have been authenticated at one network layer but are using a secure channel at a lower network layer. This term is used as a noun.

- Channel bindings: [See historical note below.]

  Generally, some data that "names" a channel or one or both of its end-points such that if this data can be shown, at a higher network layer, to be the same at both ends of a channel, then there are no MITMs between the two end-points at that higher network layer. This term is used as a noun.

  More formally, there are two types of channel bindings:

  + unique channel bindings:

    channel bindings that name a channel in a cryptographically secure manner and uniquely in time;

  + end-point channel bindings:

    channel bindings that name the authenticated end-points, or even a single end-point, of a channel which are, in turn, securely bound to the channel, but which do not identify a channel uniquely in time.

- Cryptographic binding: (e.g., "cryptographically bound") a cryptographic operation that causes an object, such as a private encryption or signing key, or an established secure channel, to "speak for" [Lampson91] some principal, such as a user, a computer, etcetera. For example, a Public Key Infrastructure for X.509 Certificates (PKIX) certificate binds a private key to the name of a principal in the trust domain of the certificate's issuer such that a possessor of said private key can act on behalf of the user (or other entity) named by the certificate.

  Cryptographic bindings are generally asymmetric in nature (not to be confused with symmetric or asymmetric key cryptography) in that an object is rendered capable of standing for another, but the reverse is not usually the case (we don't say that a user speaks for their private keys, but we do say that the user's private keys speak for the user).

Note that there may be many instances of "cryptographic binding" in an application of channel binding. The credentials that authenticate principals at the application layer bind private or secret keys to the identities of those principals, such that said keys speak for

them. A secure channel typically consists of symmetric session keys used to provide confidentiality and integrity protection to data sent over the channel; each end-point's session keys speak for that end-point of the channel. Finally, each end-point of a channel bound to authentication at the application layer speaks for the principal authenticated at the application layer on the same side of the channel.

The terms defined above have been in use for many years and have been taken to mean, at least in some contexts, what is stated below. Unfortunately this means that "channel binding" can refer to the channel binding operation and, sometimes to the name of a channel, and "channel bindings" -- a difference of only one letter -- generally refers to the name of a channel.

Note that the Extensible Authentication Protocol (EAP) [RFC3748] uses "channel binding" to refer to a facility that may appear to be similar to the one decribed here, but it is, in fact, quite different. See Section 2.2 for mode details.

2.1. Properties of Channel Binding

Applications, authentication frameworks (e.g., the GSS-API, SASL), security mechanisms (e.g., the Kerberos V GSS-API mechanism [RFC1964]), and secure channels must meet the requirements and should follow the recommendations that are listed below.

Requirements:

- In order to use channel binding, applications MUST verify that the same channel bindings are observed at either side of the channel. To do this, the application MUST use an authentication protocol at the application layer to authenticate one, the other, or both application peers (one at each end of the channel).
  - If the authentication protocol used by the application supports channel binding, the application SHOULD use it.
  - An authentication protocol that supports channel binding MUST provide an input slot in its API for a "handle" to the channel, or its channel bindings.
  - If the authentication protocol does not support a channel binding operation, but provides a "security layer" with at least integrity protection, then the application MUST use the authentication protocol's integrity protection facilities to exchange channel bindings, or cryptographic hashes thereof.

    * The name of the type of channel binding MUST be used by the application and/or authentication protocol to avoid ambiguity about which of several possible types of channels is being bound. If nested instances of the same type of channel are available, then the innermost channel MUST be used.

- Specifications of channel bindings for any secure channels MUST provide for a single, canonical octet string encoding of the channel bindings. Under this framework, channel bindings MUST start with the channel binding unique prefix followed by a colon (ASCII 0x3A).

- The channel bindings for a given type of secure channel MUST be constructed in such a way that an MITM could not easily force the channel bindings of a given channel to match those of another.

- Unique channel bindings MUST bind not only the key exchange for the secure channel, but also any negotiations and authentication that may have taken place to establish the channel.

- End-point channel bindings MUST be bound into the secure channel and all its negotiations. For example, a public key as an end-point channel binding should be used to verify a signature of such negotiations (or to encrypt them), including the initial key exchange and negotiation messages for that channel -- such a key would then be bound into the channel. A certificate name as end-point channel binding could also be bound into the channel in a similar way, though in the case of a certificate name, the binding also depends on the strength of the authentication of that name (that is, the validation of the certificate, the trust anchors, the algorithms used in the certificate path construction and validation, etcetera).

- End-point channel bindings MAY be identifiers (e.g., certificate names) that must be authenticated through some infrastructure, such as a public key infrastructure (PKI). In such cases, applications MUST ensure that the channel provides adequate authentication of such identifiers (e.g., that the certificate validation policy and trust anchors used by the channel satisfy the application's requirements). To avoid implementation difficulties in addressing this requirement, applications SHOULD use cryptographic quantities as end-point channel bindings, such as certificate-subject public keys.

- Applications that desire confidentiality protection MUST use application-layer session protection services for confidentiality protection when the bound channel does not provide confidentiality protection.

- The integrity of a secure channel MUST NOT be weakened should their channel bindings be revealed to an attacker. That is, the construction of the channel bindings for any type of secure channel MUST NOT leak secret information about the channel. End-point channel bindings, however, MAY leak information about the end-points of the channel (e.g., their names).

- The channel binding operation MUST be at least integrity protected in the security mechanism used at the application layer.

- Authentication frameworks and mechanisms that support channel binding MUST communicate channel binding failure to applications.

- Applications MUST NOT send sensitive information, requiring confidentiality protection, over the underlying channel prior to completing the channel binding operation.

Recommendations:

- End-point channel bindings where the end-points are meaningful names SHOULD NOT be used when the channel does not provide confidentiality protection and privacy protection is desired. Alternatively, channels that export such channel bindings SHOULD provide for the use of a digest and SHOULD NOT introduce new digest/hash agility problems as a result.

Options:

- Authentication frameworks and mechanisms that support channel binding MAY fail to establish authentication if channel binding fails.

- Applications MAY send information over the underlying channel and without integrity protection from the application-layer authentication protocol prior to completing the channel binding operation if such information requires only integrity protection. This could be useful for optimistic negotiations.

- A security mechanism MAY exchange integrity-protected channel bindings.

- A security mechanism MAY exchange integrity-protected digests of channel bindings. Such mechanisms SHOULD provide for hash/digest agility.

- A security mechanism MAY use channel bindings in key exchange, authentication, or key derivation, prior to the exchange of "authenticator" messages.

2.2. EAP Channel Binding

This section is informative. This document does not update EAP [RFC3748], it neither normatively describes, nor does it impose requirements on any aspect of EAP or EAP methods.

EAP [RFC3748] includes a concept of channel binding described as follows:

> The communication within an EAP method of integrity-protected channel properties such as endpoint identifiers which can be compared to values communicated via out of band mechanisms (such as via a AAA or lower layer protocol).

Section 7.15 of [RFC3748] describes the problem as one where a Network Access Server (NAS) (a.k.a. "authenticator") may lie to the peer (client) and cause the peer to make incorrect authorization decisions (e.g., as to what traffic may transit through the NAS). This is not quite like the purpose of generic channel binding (MITM detection).

Section 7.15 of [RFC3748] calls for "a protected exchange of channel properties such as endpoint identifiers" such that "it is possible to match the channel properties provided by the authenticator via out-of-band mechanisms against those exchanged within the EAP method".

This has sometimes been taken to be very similar to the generic notion of channel binding provided here. However, there is a very subtle difference between the two concepts of channel binding that makes it much too difficult to put forth requirements and recommendations that apply to both. The difference is about the lower-layer channel:

- In the generic channel binding case, the identities of either end of this channel are irrelevant to anything other than the construction of a name for that channel, in which case the identities of the channel's end-points must be established a priori.

- Whereas in the EAP case, the identity of the NAS end of the channel, and even security properties of the channel itself, may be established during or after authentication of the EAP peer to the EAP server.

In other words: there is a fundamental difference in mechanics (timing of lower-layer channel establishment) and in purpose (authentication of lower-layer channel properties for authorization purposes vs. MITM detection).

After some discussion we have concluded that there is no simple way to obtain requirements and recommendations that apply to both generic and EAP channel binding. Therefore, EAP is out of the scope of this document.

3. Authentication and Channel Binding Semantics

Some authentication frameworks and/or mechanisms provide for channel binding, such as the GSS-API and some GSS-API mechanisms, whereas others may not, such as SASL (however, ongoing work is adding channel binding support to SASL). Semantics may vary with respect to negotiation, how the binding occurs, and handling of channel binding failure (see below).

Where suitable channel binding facilities are not provided, application protocols MAY include a separate, protected exchange of channel bindings. In order to do this, the application-layer authentication service must provide message protection services (at least integrity protection).

3.1. The GSS-API and Channel Binding

The GSS-API [RFC2743] provides for the use of channel binding during initialization of GSS-API security contexts, though GSS-API mechanisms are not required to support this facility.

This channel binding facility is described in [RFC2743] and [RFC2744].

GSS-API mechanisms must fail security context establishment when channel binding fails, and the GSS-API provides no mechanism for the negotiation of channel binding. As a result GSS-API applications must agree a priori, through negotiation or otherwise, on the use of channel binding.

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 binding within their existing methods for negotiating GSS-API mechanisms. For example, NFSv4 [RFC3530] provides its own GSS-API mechanism negotiation, as does the SSHv2 protocol [RFC4462]. Such pseudo-mechanisms are being proposed separately, see [STACKABLE].

3.2. SASL and Channel Binding

SASL [RFC4422] does not yet provide for the use of channel binding during initialization of SASL contexts.

Work is ongoing [SASL-GS2] to specify how SASL, particularly its new bridge to the GSS-API, performs channel binding. SASL will likely differ from the GSS-API in its handling of channel binding failure (i.e., when there may be an MITM) in that channel binding success/failure will only affect the negotiation of SASL security layers. That is, when channel binding succeeds, SASL should select no security layers, leaving session cryptographic protection to the secure channel that SASL authentication has been bound to.

4. Channel Bindings Specifications

Channel bindings for various types of secure channels are not described herein. Some channel bindings specifications can be found in:

Secure Channel Type	Reference
SSHv2	[SSH-CB]
TLS	[TLS-CB]
IPsec	There is no specification for IPsec channel bindings yet, but the IETF Better Than Nothing Security (BTNS) WG is working to specify IPsec channels, and possibly IPsec channel bindings.

4.1. Examples of Unique Channel Bindings

The following text is not normative, but is here to show how one might construct channel bindings for various types of secure channels.

For SSHv2 [RFC4251] the SSHv2 session ID should suffice as it is a cryptographic binding of all relevant SSHv2 connection parameters: key exchange and negotiation.

The TLS [RFC4346] session ID is simply assigned by the server. As such, the TLS session ID does not have the required properties to be useful as a channel binding because any MITM, posing as the server,

can simply assign the same session ID to the victim client as the server assigned to the MITM. Instead, the initial, unencrypted TLS finished messages (the client's, the server's, or both) are sufficient as they are the output of the TLS pseudo-random function, keyed with the session key, applied to all handshake material.

4.2. Examples of End-Point Channel Bindings

The following text is not normative, but is here to show how one might construct channel bindings for various types of secure channels.

For SSHv2 [RFC4251] the SSHv2 host public key, when present, should suffice as it is used to sign the algorithm suite negotiation and Diffie-Hellman key exchange; as long the client observes the host public key that corresponds to the private host key that the server used, then there cannot be an MITM in the SSHv2 connection. Note that not all SSHv2 key exchanges use host public keys; therefore, this channel bindings construction is not as useful as the one given in Section 4.1.

For TLS [RFC4346]the server certificate should suffice for the same reasons as above. Again, not all TLS cipher suites involve server certificates; therefore, the utility of this construction of channel bindings is limited to scenarios where server certificates are commonly used.

5. Uses of Channel Binding

Uses for channel binding identified so far:

- Delegating session cryptographic protection to layers where hardware can reasonably be expected to support relevant cryptographic protocols:
  - NFSv4 [RFC3530] with Remote Direct Data Placement (RDDP) [NFS-DDP] for zero-copy reception where network interface controllers (NICs) support RDDP. Cryptographic session protection would be delegated to Encapsulating Security Payload (ESP) [RFC4303] / Authentication Headers (AHs) [RFC4302].
  - iSCSI [RFC3720] with Remote Direct Memory Access (RDMA) [RFC5046]. Cryptographic session protection would be delegated to ESP/AH.
  - HTTP with TLS [RFC2817] [RFC2818]. In situations involving proxies, users may want to bind authentication to a TLS channel between the last client-side proxy and the first server-side

proxy ("concentrator"). There is ongoing work to expand the set of choices for end-to-end authentication at the HTTP layer, that, coupled with channel binding to TLS, would allow for proxies while not forgoing protection over public internets.

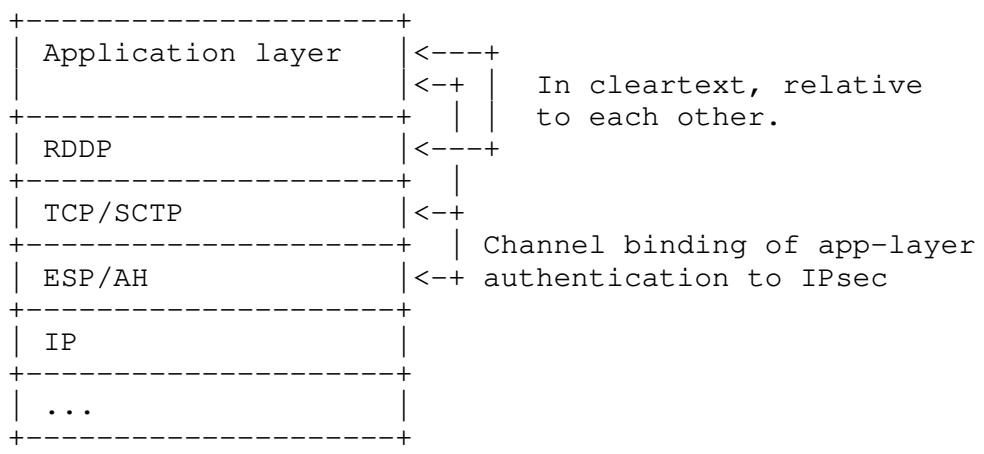
- Reducing the number of live cryptographic contexts that an application must maintain:

  * NFSv4 [RFC3530] multiplexes multiple users onto individual connections. Each user is authenticated separately, and users' remote procedure calls (RPCs) are protected with per-user GSS-API security contexts. This means that large timesharing clients must often maintain many cryptographic contexts per-NFSv4 connection. With channel binding to IPsec, they could maintain a much smaller number of cryptographic contexts per-NFSv4 connection, thus reducing memory pressure and interactions with cryptographic hardware.

For example, applications that wish to use RDDP to achieve zero-copy semantics on reception may use a network layer understood by NICs to offload delivery of application data into pre-arranged memory buffers. Note that in order to obtain zero-copy reception semantics either application data has to be in cleartext relative to this RDDP layer, or the RDDP implementation must know how to implement cryptographic session protection protocols used at the application layer.

There are a multitude of application-layer cryptographic session protection protocols available. It is not reasonable to expect that NICs should support many such protocols. Further, some application protocols may maintain many cryptographic session contexts per-connection (for example, NFSv4 does). It is thought to be simpler to push the cryptographic session protection down the network stack (to IPsec), and yet be able to produce NICs that offload other operations (i.e., TCP/IP, ESP/AH, and DDP), than it would be to add support in the NIC for the many session cryptographic protection protocols in use in common applications at the application layer.

The following figure shows how the various network layers are related:



6. Benefits of Channel Binding to Secure Channels

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

- Performance improvements by avoiding double protection of application data 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.

  Where protocols layered above RDDP use privacy protection, RDDP offload cannot be done. Thus, by using channel binding 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 with RPCSEC_GSS [RFC2203].

Delegation of session cryptographic protection to IPsec requires features not yet specified. There is ongoing work to specify:

- IPsec channels [CONN-LATCH];

- Application programming interfaces (APIs) related to IPsec channels [BTNS-IPSEC];
- Channel bindings for IPsec channels;
- Low infrastructure IPsec authentication [BTNS-CORE].

7. IANA Considerations

IANA has created a new registry for channel bindings specifications for various types of channels.

The purpose of this registry is not only to ensure uniqueness of values used to name channel bindings, but also to provide a definitive reference to technical specifications detailing each channel binding available for use on the Internet.

There is no naming convention for channel bindings: any string composed of US-ASCII alphanumeric characters, period ('.'), and dash ('-') will suffice.

The procedure detailed in Section 7.1 is to be used for registration of a value naming a specific individual mechanism.

7.1. Registration Procedure

Registration of a new channel binding requires expert review as defined in BCP 26 [RFC2434].

Registration of a channel binding is requested by filling in the following template:

- Subject: Registration of channel binding X
- Channel binding unique prefix (name):
- Channel binding type: (One of "unique" or "end-point")
- Channel type: (e.g., TLS, IPsec, SSH, etc.)
- Published specification (recommended, optional):
- Channel binding is secret (requires confidentiality protection): yes/no
- Description (optional if a specification is given; required if no published specification is specified):

- Intended usage: (one of COMMON, LIMITED USE, or OBSOLETE)
- Person and email address to contact for further information:
- Owner/Change controller name and email address:
- Expert reviewer name and contact information: (leave blank)
- Note: (Any other information that the author deems relevant may be added here.)

and sending it via electronic mail to <channel-binding@ietf.org> (a public mailing list) and carbon copying IANA at <iana@iana.org>. After allowing two weeks for community input on the mailing list to be determined, an expert will determine the appropriateness of the registration request and either approve or disapprove the request with notice to the requestor, the mailing list, and IANA.

If the expert approves registration, it adds her/his name to the submitted registration.

The expert has the primary responsibility of making sure that channel bindings for IETF specifications go through the IETF consensus process and that prefixes are unique.

The review should focus on the appropriateness of the requested channel binding for the proposed use, the appropriateness of the proposed prefix, and correctness of the channel binding type in the registration. The scope of this request review may entail consideration of relevant aspects of any provided technical specification, such as their IANA Considerations section. However, this review is narrowly focused on the appropriateness of the requested registration and not on the overall soundness of any provided technical specification.

Authors are encouraged to pursue community review by posting the technical specification as an Internet-Draft and soliciting comment by posting to appropriate IETF mailing lists.

7.2. Comments on Channel Bindings Registrations

Comments on registered channel bindings should first be sent to the "owner" of the channel bindings and to the channel binding mailing list.

Submitters of comments may, after a reasonable attempt to contact the owner, request IANA to attach their comment to the channel binding type registration itself by sending mail to <iana@iana.org>. At

IANA's sole discretion, IANA may attach the comment to the channel bindings registration.

7.3. Change Control

Once a channel bindings registration has been published by IANA, the author may request a change to its definition. The change request follows the same procedure as the registration request.

The owner of a channel bindings may pass responsibility for the channel bindings to another person or agency by informing IANA; this can be done without discussion or review.

The IESG may reassign responsibility for a channel bindings registration. The most common case of this will be to enable changes to be made to mechanisms where the author of the registration has died, has moved out of contact, or is otherwise unable to make changes that are important to the community.

Channel bindings registrations may not be deleted; mechanisms that are no longer believed appropriate for use can be declared OBSOLETE by a change to their "intended usage" field. Such channel bindings will be clearly marked in the lists published by IANA.

The IESG is considered to be the owner of all channel bindings that are on the IETF standards track.

8. Security Considerations

Security considerations appear throughout this document. In particular see Section 2.1.

When delegating session protection from one layer to another, one will almost certainly be making some session security trade-offs, such as using weaker cipher modes in one layer than might be used in the other. Evaluation and comparison of the relative cryptographic strengths of these is difficult, may not be easily automated, and is far out of scope for this document. Implementors and administrators should understand these trade-offs. Interfaces to secure channels and application-layer authentication frameworks and mechanisms could provide some notion of security profile so that applications may avoid delegation of session protection to channels that are too weak to match a required security profile.

Channel binding makes "anonymous" channels (where neither end-point is strongly authenticated to the other) useful. Implementors should avoid making it easy to use such channels without channel binding.

The security of channel binding depends on the security of the channels, the construction of their channel bindings, and the security of the authentication mechanism used by the application and its channel binding method.

Channel bindings should be constructed in such a way that revealing the channel bindings of a channel to third parties does not weaken the security of the channel. However, for end-point channel bindings disclosure of the channel bindings may disclose the identities of the peers.

8.1. Non-Unique Channel Bindings and Channel Binding Re-Establishment

Application developers may be tempted to use non-unique channel bindings for fast re-authentication following channel re-establishment. Care must be taken to avoid the possibility of attacks on multi-user systems.

Consider a user multiplexing protocol like NFSv4 using channel binding to IPsec on a multi-user client. If another user can connect directly to port 2049 (NFS) on some server using IPsec and merely assert RPCSEC_GSS credential handles, then this user will be able to impersonate any user authenticated by the client to the server. This is because the new connection will have the same channel bindings as the NFS client's! To prevent this, the server must require that at least a host-based client principal, and perhaps all the client's user principals, re-authenticate and perform channel binding before the server will allow the clients to assert RPCSEC_GSS context handles. Alternatively, the protocol could require a) that secure channels provide confidentiality protection and b) that fast re-authentication cookies be difficult to guess (e.g., large numbers selected randomly).

In other contexts there may not be such problems, for example, in the case of application protocols that don't multiplex users over a single channel and where confidentiality protection is always used in the secure channel.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

9.2. Informative References

[BTNS-AS] Touch, J., Black, D., and Y. Wang, "Problem and Applicability Statement for Better Than Nothing Security (BTNS)", Work in Progress, October 2007.

[BTNS-CORE] Richardson, M. and N. Williams, "Better-Than-Nothing-Security: An Unauthenticated Mode of IPsec", Work in Progress, September 2007.

[BTNS-IPSEC] Richardson, M. and B. Sommerfeld, "Requirements for an IPsec API", Work in Progress, April 2006.

[CONN-LATCH] Williams, N., "IPsec Channels: Connection Latching", Work in Progress, September 2007.

[Lampson91] Lampson, B., Abadi, M., Burrows, M., and E. Wobber, "Authentication in Distributed Systems: Theory and Practive", October 1991.

[NFS-DDP] Callaghan, B. and T. Talpey, "NFS Direct Data Placement", Work in Progress, July 2007.

[RFC1964] Linn, J., "The Kerberos Version 5 GSS-API Mechanism", RFC 1964, June 1996.

[RFC2203] Eisler, M., Chiu, A., and L. Ling, "RPCSEC_GSS Protocol Specification", RFC 2203, September 1997.

[RFC2401] Kent, S. and R. Atkinson, "Security Architecture for the Internet Protocol", RFC 2401, November 1998.

[RFC2434] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 2434, October 1998.

[RFC2743] Linn, J., "Generic Security Service Application Program Interface Version 2, Update 1", RFC 2743, January 2000.

[RFC2744] Wray, J., "Generic Security Service API Version 2 : C-bindings", RFC 2744, January 2000.

[RFC2817] Khare, R. and S. Lawrence, "Upgrading to TLS Within HTTP/1.1", RFC 2817, May 2000.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000.

[RFC3530] Shepler, S., Callaghan, B., Robinson, D., Thurlow, R., Beame, C., Eisler, M., and D. Noveck, "Network File System (NFS) version 4 Protocol", RFC 3530, April 2003.

[RFC3720] Satran, J., Meth, K., Sapuntzakis, C., Chadalapaka, M., and E. Zeidner, "Internet Small Computer Systems Interface (iSCSI)", RFC 3720, April 2004.

[RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, "Extensible Authentication Protocol (EAP)", RFC 3748, June 2004.

[RFC4120] Neuman, C., Yu, T., Hartman, S., and K. Raeburn, "The Kerberos Network Authentication Service (V5)", RFC 4120, July 2005.

[RFC4251] Ylonen, T. and C. Lonvick, "The Secure Shell (SSH) Protocol Architecture", RFC 4251, January 2006.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.

[RFC4302] Kent, S., "IP Authentication Header", RFC 4302, December 2005.

[RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, December 2005.

[RFC4346] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1", RFC 4346, April 2006.

[RFC4422] Melnikov, A. and K. Zeilenga, "Simple Authentication and Security Layer (SASL)", RFC 4422, June 2006.

[RFC4462] Hutzelman, J., Salowey, J., Galbraith, J., and V. Welch, "Generic Security Service Application Program Interface (GSS-API) Authentication and Key Exchange for the Secure Shell (SSH) Protocol", RFC 4462, May 2006.

[RFC5046] Ko, M., Chadalapaka, M., Hufferd, J., Elzur, U., Shah, H., and P. Thaler, "Internet Small Computer System Interface (iSCSI) Extensions for Remote Direct Memory Access (RDMA)", RFC 5046, October 2007.

[SASL-GS2] Josefsson, S., "Using GSS-API Mechanisms in SASL: The GS2 Mechanism Family", Work in Progress, October 2007.

[SSH-CB] Williams, N., "Channel Binding Identifiers for Secure Shell Channels", Work in Progress, November 2007.

[STACKABLE] Williams, N., "Stackable Generic Security Service Pseudo-Mechanisms", Work in Progress, June 2006.

[TLS-CB] Altman, J. and N. Williams, "Unique Channel Bindings for TLS", Work in Progress, November 2007.

Appendix A. Acknowledgments

Thanks to Mike Eisler for his work on the Channel Conjunction Mechanism document and for bringing the problem to a head, Sam Hartman for pointing out that channel binding provides a general solution to the channel binding problem, and Jeff Altman for his suggestion of using the TLS finished messages as the TLS channel bindings. Also, thanks to Bill Sommerfeld, Radia Perlman, Simon Josefsson, Joe Salowey, Eric Rescorla, Michael Richardson, Bernard Aboba, Tom Petch, Mark Brown, and many others.

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