

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 8, 2011

N. Cam-Winget
J. Salowey
H. Zhou
Cisco Systems
March 7, 2011

Transport Layer Security (TLS) Based Transports for Network Endpoint
Assessment (NEA) Protocol Exchanges
draft-cam-winget-eap-tlv-03

Abstract

This document describes how Network Endpoint Assessment (NEA) data can be carried under the protection of a Transport Layer Security (TLS) secured tunnel. This document defines NEA transports for TLS-based Extensible Authentication Protocol (EAP) tunnel methods and for TLS used over TCP.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

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."

This Internet-Draft will expire on September 8, 2011.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Introduction	4
2. Specification Requirements	4
3. Protocol Layering Model	4
4. Protocol Flows	5
4.1. PT-TCP Protocol Flow	5
4.1.1. Initiating a PT-TCP session	5
4.1.2. TCP Port Usage	5
4.1.3. TLS Setup Phase	5
4.1.4. NEA Data Transport Phase in PT-TCP	6
4.1.5. Entity Authentication using SASL in PT-TCP	6
4.1.5.1. Service Name	7
4.1.5.2. Mechanism Negotiation	7
4.1.5.3. Message Definition	7
4.1.5.4. Authorization Identity	7
4.1.5.5. Aborting Authentication	7
4.1.5.6. Security Layers	7
4.1.5.7. Multiple Authentications	7
4.2. Tunnel EAP Message Flow	8
5. Packet Formats	8
5.1. PT-TCP transport Format	9
5.1.1. NEA TLV	10
5.1.2. SASL-MECH TLV	11
5.1.3. SASL-AUTH TLV	12
5.1.4. SASL-RESULT TLV	13
5.2. Using tunnel EAP to transport NEA data	14
5.2.1. Carrying NEA data in PEAP or EAP-FAST	14
5.2.2. Carrying NEA data in TTLS	16
6. Binding the PA exchange to the TLS Tunnel	17
7. Security Considerations	17
8. IANA Considerations	17

9. Acknowledgements	18
10. References	18
10.1. Normative References	18
10.2. Informative References	18
Appendix A. Evaluation Against NEA Requirements	19
A.1. Evaluation Against Requirement C-1	19
A.2. Evaluation Against Requirement C-2	19
A.3. Evaluation Against Requirement C-3	19
A.4. Evaluation Against Requirement C-4	20
A.5. Evaluation Against Requirement C-5	20
A.6. Evaluation Against Requirement C-6	21
A.7. Evaluation Against Requirement C-7	21
A.8. Evaluation Against Requirement C-8	21
A.9. Evaluation Against Requirement C-9	22
A.10. Evaluation Against Requirement C-10	22
A.11. Evaluation Against Requirement C-11	22
A.12. Evaluation Against Requirement PT-1	23
A.13. Evaluation Against Requirement PT-2	23
A.14. Evaluation Against Requirement PT-3	23
A.15. Evaluation Against Requirement PT-4	24
A.16. Evaluation Against Requirement PT-5	24
A.17. Evaluation Against Requirement PT-6	24
A.18. Evaluation Against Requirement PT-7	25
A.19. Evaluation Against Requirement PT-8	25
A.20. Evaluation Against Requirement PT-9	25
Authors' Addresses	25

1. Introduction

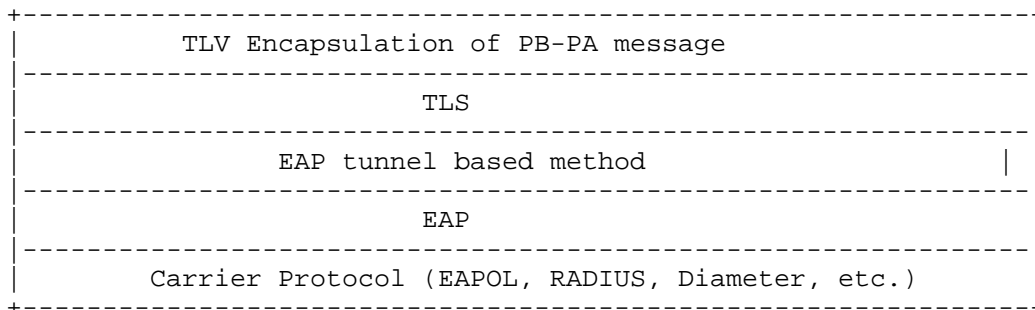
NEA has standardized a transport agnostic Posture Broker (PB) protocol defined in [RFC5793] to effect a network endpoint assessment between a Posture Broker Client and a Posture Broker Server. These PB messages can be transported inside the already defined Type-Length-Value containers in existing TLS-based tunnel EAP methods such as PEAP, EAP-FAST and TTLS. Similarly, this document also defines a TCP based transport, PT-TCP, that uses TLVs encapsulated within TLS.

2. Specification Requirements

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] .

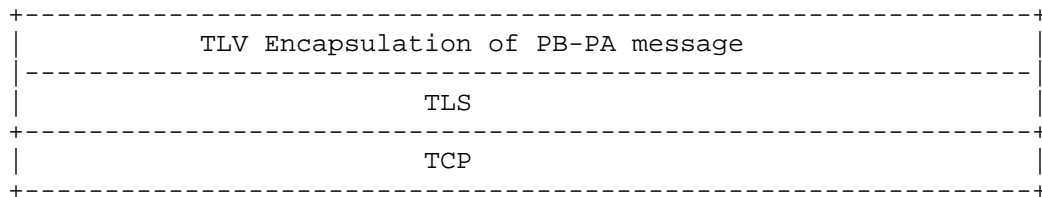
3. Protocol Layering Model

When using EAP as the transport, the PB messages can be encapsulated in the TLVs defined by the tunnel EAP methods. For TLS a new TLV container is defined to facilitate the PB transport over TCP. The following diagram demonstrates the relationship between protocols when an EAP tunnel method is used:



EAP based Protocol Layering Model

The following diagram demonstrates the protocol relationship of PB when PT-TCP is used:



PT-TCP based Protocol Layering Model

4. Protocol Flows

There are two distinct phases in TLS based transport operation:

1. **TLS Setup Phase:** are the messages used to establish TLS channel protection for the posture messages. The TLS Setup Phase begins with either the Posture Transport Client or Posture Transport Server initiating the TLS Handshake protocol to establish the protected TLS channel.
2. **Data Transport Phase:** are the messages that are protected by the TLS Record encapsulation. This phase is usually broken up into an optional entity authentication phase followed by the exchange of TLVs carrying NEA data.

4.1. PT-TCP Protocol Flow

This section describes the general flow of messages between the NEA Posture Transport Client and the NEA Posture Transport Server.

4.1.1. Initiating a PT-TCP session

With the use of TLS as the transport, it is possible for either the Posture Transport Client or the Posture Transport Server to initiate a PT-TCP session.

4.1.2. TCP Port Usage

IANA is requested to allocate a TCP Port number for the use of PT-TCP so that both the Posture Transport Client and Posture Transport Server can communicate on a known allocated port.

4.1.3. TLS Setup Phase

Typically, it is the NEA Client (e.g. the Posture Transport Client) that initiates the TLS Setup Phase. However, either party, e.g. the

Posture Transport Client or the Posture Transport Server may establish a TCP connection and initiate the TLS Handshake protocol. Furthermore, the TLS Handshake protocol is also used to establish the cryptographic protections used to secure the data carried within TLS Records.

In typical deployments, it is expected for the initiator of a NEA exchange to initiate the TLS Setup. However, this specification allows for multiple NEA data transactions and as such, each transaction may originate from either the NEA client or the NEA server. Furthermore, through the use of the TLS session capabilities, PT-TCP also allows for the re-use of the TLS based (PT-TCP) session to allow either the NEA Client or the NEA Server to trigger subsequent NEA exchanges.

4.1.4. NEA Data Transport Phase in PT-TCP

Once the PT-TCP session has been established, either the NEA Client or the NEA Server can trigger a NEA data transaction (typically a posture assessment). The initiator for the NEA data transaction encapsulates the PB messages in a TLV as described in Section 5.1.

As PT-TCP is full-duplex (by the TLS design), it supports the full capabilities of the PB-TNC state machine.

4.1.5. Entity Authentication using SASL in PT-TCP

Implementations may support entity authentication through the use of SASL [RFC4422]. This section details the SASL profile for PT-TCP.

Typically, the PT-TCP initiator will also initiate the SASL exchange. The responder presents a list of SASL mechanism it supports through the use of the SASL-AUTH-MECH TLV. The initiator may request a list of SASL authentication mechanisms by sending an empty list of mechanisms in the SASL-AUTH-MECH TLV.

The initiator starts the authentication by sending a SASL-AUTH TLV with the mech field containing the name of the mechanism it selects. If the selected mechanism has an initial response then the client includes that response in the auth-data field. If necessary the responder sends a SASL-AUTH TLV with the auth-data field containing a SASL challenge for the selected mechanism. The SASL-AUTH exchange continues in this manner until the authentication completes upon completion the responder sends a SASL-RESULT TLV. If the authentication is successful the SASL-RESULT TLV contains a result code of success. If the authentication fails the SASL-RESULT TLV contains a result code indicating the reason for the failure. The initiator may abort the exchange by sending a SASL-RESULT TLV with an

ABORT result code.

Implementations MUST provide the EXTERNAL SASL mechanism if the initiator is authenticated during the TLS establishment. Implementations MUST also support the PLAIN SASL mechanism.

4.1.5.1. Service Name

The service name for PT-TCP is "nea-pt-tcp".

4.1.5.2. Mechanism Negotiation

Mechanism Negotiation is performed using the SASL-AUTH-MECH TLV. The SASL-AUTH-MECH TLV contains the list of mechanisms supported by the responder. The initiator may send a SASL-AUTH-MECH TLV with an empty list to request a list format from the responder.

4.1.5.3. Message Definition

The initiator starts authentication by sending a SASL-AUTH TLV indicating the selected mechanism. The initial message may contain an initial response if required by the selected mechanism. Subsequent challenges and response are carried within SASL-AUTH TLVs between the initiator and responder carrying the authentication data for the mechanism. The authentication outcome is communicated in a SASL-RESULT TLV containing a status code.

4.1.5.4. Authorization Identity

The nea-pt-tcp protocol does not make use of an authorization identity.

4.1.5.5. Aborting Authentication

The initiator may abort the authentication exchange by sending the SASL-AUTH-RESULT TLV with a status code of ABORT.

4.1.5.6. Security Layers

The NEA PT-TCP protocol always runs under the protection of TLS. SASL security layers are not used.

4.1.5.7. Multiple Authentications

Only one authentication may be in progress at any one time. Once an authentication completes, successfully or unsuccessfully, a new authentication may be initiated.

4.2. Tunnel EAP Message Flow

This section discusses the general flow of messages between the NEA Client's Posture Transport Client and the NEA Server's Posture Transport Server in order to perform NEA assessments when using a tunnel EAP method.

When NEA data exchange is conducted in a tunnel EAP method, it typically consists of four phases:

1. Establishment of EAP tunnel method: a secure and protected TLS channel is established between the Transport Client and Transport Server, after the Transport Server's identity has been authenticated and a shared secret encryption key is established between them.
2. Entity authentication: during this phase, the NEA Client's Posture Transport Client's identity might be optionally authenticated, so appropriate posture assessment policy could be applied according to the authenticated entity. Typically, it is done via an inner EAP method or authentication exchanges within the protected tunnel. In addition, the identity could also be authenticated as part of the tunnel establishment instead (e.g., the client sends a client certificate as part of the tunnel establishment).
3. Posture assessment: the posture data are exchanged between the NEA Client's Posture Transport client and NEA Server's Posture Transport Server. The posture data is encapsulated in a TLV or TLV like type object, as described in Section 5.2.
4. Conclusion phase: the result of the authentication and/or posture assessment is exchanged between the client and server, so they will have synchronized state. Optional cryptographic binding might be done to ensure both peers are involved in both the tunnel establishment and the inner method exchanges. Both sides are ready to tear down the tunnel and finish the EAP method.

At the end of the tunnel EAP method, an EAP-Success or EAP-Failure will be sent by the EAP server to indicate the end of the EAP authentication, and the NAS will apply appropriate authorization policy based on the authentication and posture assessment result.

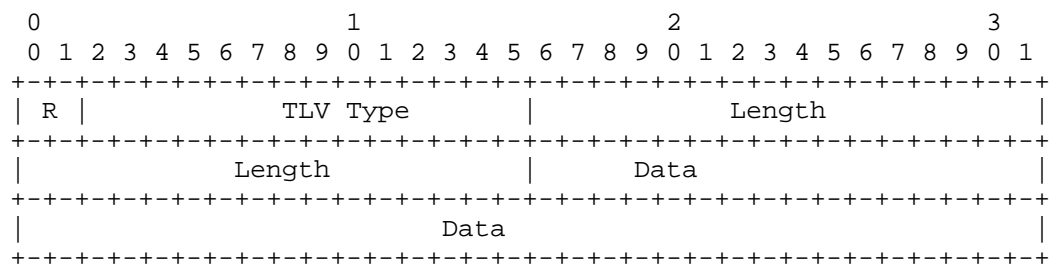
5. Packet Formats

As there is no explicit authentication expected in the PB-PA message exchanges, no new inner EAP method is required; rather, the TLV

formats defined in existing EAP tunnel methods can be used to encapsulate and transport PB-PA messages. Similarly, when using TLS a TLV format can be defined to carry NEA data. This section describes how NEA data can be carried in either a tunnel EAP method or TLS.

5.1. PT-TCP transport Format

This section defines a Type-Length-Value (TLV) encapsulation for carrying NEA data in a TLS channel. The TLS channel **MUST** be protected to carry NEA data using the encapsulation defined below. The fields are transmitted from left to right.



R

Reserved, set to zero (0)

TLV Type

TLV Type Code. Allocated Types include:

- 0 Reserved
- 1 NEA TLV
- 2 SASL-MECH TLV
- 3 SASL-AUTH TLV
- 4 SASL-RESULT TLV

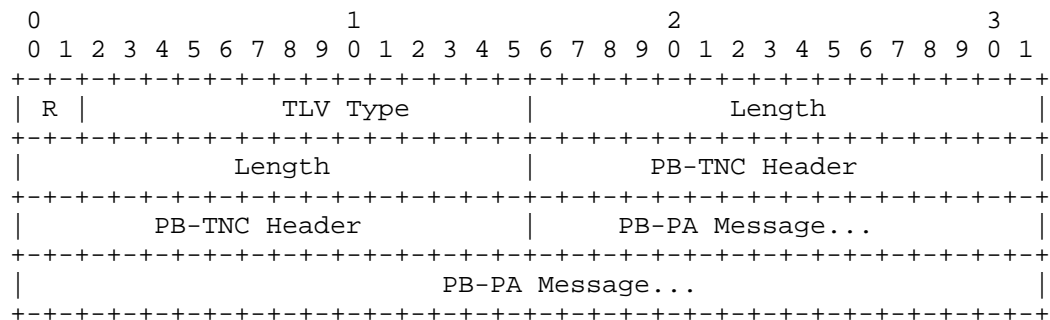
Length

The length of the Data field in octets.

Data

Data according to the TLV type.

5.1.1. NEA TLV



R

Reserved, set to zero (0)

TLV Type

1 for NEA TLV

Length

The length of the Value field in octets.

PB-TNC Header

The PB-TNC encapsulation header as described in [RFC5793].

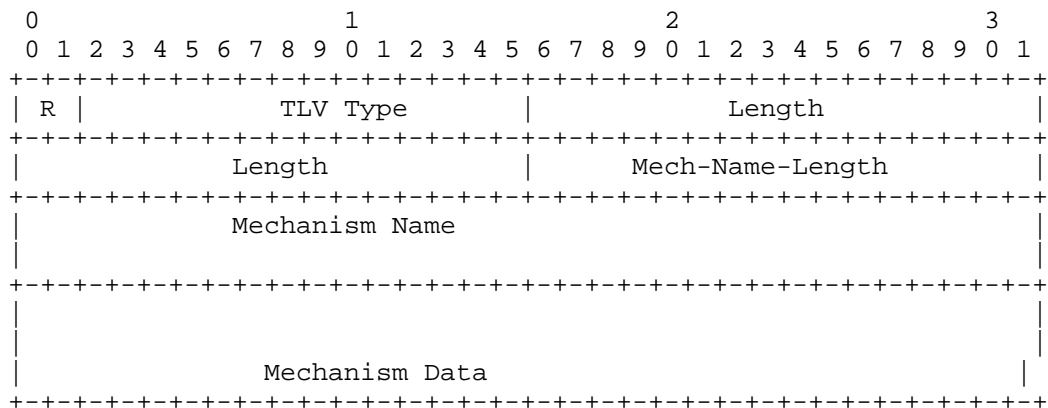
Mech-Name-Length

Length of the mechanism name in bytes.

Mech-Name

SASL mechanism Name adhering to the rules defined in [RFC4422].

5.1.3. SASL-AUTH TLV



The SASL-AUTH TLV contains data pertaining a SASL mechanism. The mechanism name is included in each SASL-AUTH TLV. The TLV is used by the initiator to select from a list of supported mechanisms provided by the responder. The initial response from the initiator may contain Mechanism Data containing the initial response. If the mechanism selected does not use an initial response then the mechanism data field is not included. The SASL-AUTH TLV is also used to communicate SASL mechanism data from the responder to the initiator.

R

Reserved, set to zero (0)

TLV Type

3 for SASL-AUTH TLV

Length

The length of the Value field in octets. The value field contains a mechanism name and optional mechanism data..

Mech-Name-Length

Length of the mechanism name in bytes.

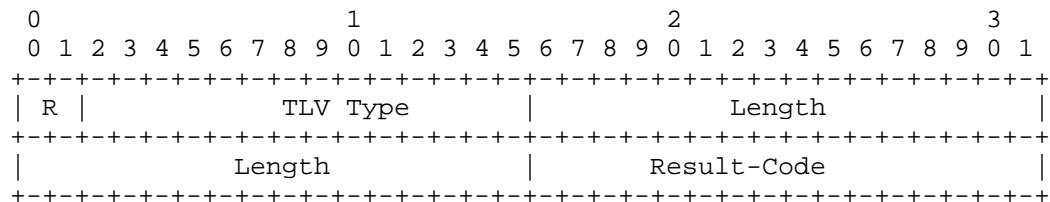
Mech-Name

SASL mechanism Name adhering to the rules defined in [RFC4422]. This is the mechanism selected for use by the initiator.

Mech-Data

SASL mechanism data for named mechanism. This field may be omitted in the initial response from the initiator if the selected mechanism does not use an initial response.

5.1.4. SASL-RESULT TLV



The SASL-RESULT TLV contains the result of the SASL Exchange. A result code of 0 indicates success. Other result codes indicate some sort of failure. A result code of 1 indicates the exchange was aborted by the sender. A result code of 2 indicates a failure within the mechanism. Only the responder side of the conversation may send a successful result code. Either side may send a failure result code which terminates the current authentication conversation.

R

Reserved, set to zero (0)

TLV Type

4 for SASL-Result TLV

Length

The length of the Value field in octets. This field is set to 2.

Result Code

The value of the result code.

0 Success

1 Abort

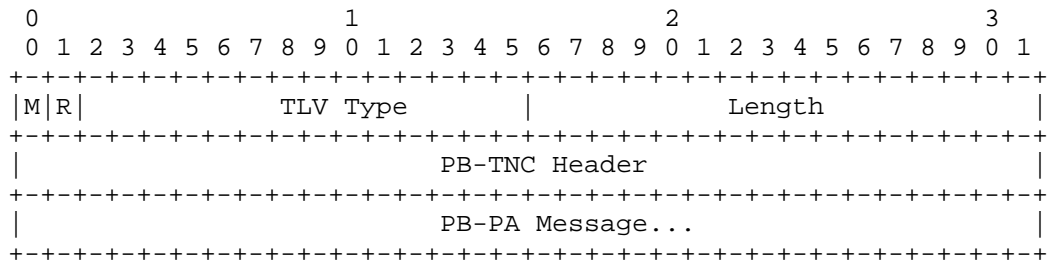
2 Mechanism Failure

5.2. Using tunnel EAP to transport NEA data

This section describes the TLV encapsulation used in three predominant tunnel EAP methods deployed today: PEAP, EAP-FAST and TTLS. When using EAP tunnel methods, the tunnel MUST be protected.

5.2.1. Carrying NEA data in PEAP or EAP-FAST

As TLV format for PEAP and EAP-FAST are the same, the diagram below shows how PB-PA messages can be encapsulated in the TLVs. Note however that the type assignments when using PEAP versus EAP-FAST may be different. The fields are transmitted from left to right.



M

0 Optional TLV

1 Mandatory TLV

R

Reserved, set to zero (0)

TLV Type

The EAP-FAST NEA TLV type:

TBD

Length

The length of the Value field in octets.

PB-TNC Header

The PB-TNC encapsulation header as described in [RFC5793].

PB-PA Message

The message between the Posture Broker Client and Posture Broker Server as described in [RFC5793].

6. Binding the PA exchange to the TLS Tunnel

Some implementations of the NEA system allow for the external validation of the data collected and sent by the posture collector. In these cases, an external measurement agent (EMA) signs the data sent by the collector. In order to prevent posture data of the endpoint from being used on another machine, the TLS tunnel and the posture data signed by the EMA must be bound together. This is done using the "tls-unique" channel binding defined in RFC 5929 [RFC5929]. The data from the first TLS Finished message sent on the most recent TLS connection handshake is included in the data signed by the EMA. The PA attributes used are specific to the EMA used by the posture collector.

The "tls-unique" channel-binding data can be used whenever a TLS transport is provided, including TLS over TCP and TLS used in tunnel EAP methods. It is RECOMMENDED that posture collectors that support an EMA provide a PA attribute to carry the "tls-unique" channel binding data. The channel binding data MAY be combined with other data using a cryptographic hash or similar technique. The channel binding attribute MUST be signed by the EMA. Posture validators that receive channel binding data MUST verify that it is consistent with the channel binding data obtained from the server-side of the TLS connection.

7. Security Considerations

The NEA TLV container carries network endpoint assessment information between the Posture Broker Client and the Posture Broker Server. As some of this data can be sensitive, TLS cipher suites that provide encryption are RECOMMENDED.

To address the potential man-in-the-middle attack similar to the Asokan attack described in [I-D.salowey-nea-asokan] the channel binding mechanism defined in Section 6 SHOULD be used whenever an EMA is available to sign the posture data.

8. IANA Considerations

IANA is requested to assign a TCP port number in the "Registered Port" range with the keyword "pt-tcp". This port will be the default port for PT-TCP defined in this document.

IANA is requested to allocate a TLV type from the EAP-FAST TLV Type registry for carrying posture data as specified in Section 5.2.1.

IANA is requested to allocate a Diameter AVP code from the Diameter AVP code registry for carrying posture data as specified in Section 5.2.2.

This document defines a registry for TLV types to be carried within PT-TCP, which may be assigned by Specification Required as defined in [RFC2434]

9. Acknowledgements

The authors would like to recognize Susan Thomson, Syam Appala and Subbu Srinivasan for providing input into this draft.

10. References

10.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2434] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 2434, October 1998.
- [RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowitz, "Extensible Authentication Protocol (EAP)", RFC 3748, June 2004.
- [RFC4422] Melnikov, A. and K. Zeilenga, "Simple Authentication and Security Layer (SASL)", RFC 4422, June 2006.
- [RFC4493] Song, JH., Poovendran, R., Lee, J., and T. Iwata, "The AES-CMAC Algorithm", RFC 4493, June 2006.
- [RFC5793] Sahita, R., Hanna, S., Hurst, R., and K. Narayan, "PB-TNC: A Posture Broker (PB) Protocol Compatible with Trusted Network Connect (TNC)", RFC 5793, March 2010.
- [RFC5929] Altman, J., Williams, N., and L. Zhu, "Channel Bindings for TLS", RFC 5929, July 2010.

10.2. Informative References

- [I-D.salowey-nea-asokan]
Salowey, J. and S. Hanna, "NEA Asokan Attack Analysis", draft-salowey-nea-asokan-00 (work in progress),

October 2010.

Appendix A. Evaluation Against NEA Requirements

This section evaluates both the PT-TCP and EAP based protocols against the PT requirements defined in the NEA Overview and Requirements and PB-TNC specifications.

A.1. Evaluation Against Requirement C-1

Requirement C-1 states:

C-1 NEA protocols MUST support multiple round trips between the NEA Client and the NEA Server in a single assessment.

PT-TCP meets this requirement. By using the TLS protocol over TCP, multiple roundtrips of TLS records and thus PT-TCP messages are allowed.

Tunnel EAP meets this requirement. All available Tunnel EAP methods are based on the TLS design which allows for multiple round trips.

A.2. Evaluation Against Requirement C-2

Requirement C-2 states:

C-2 NEA protocols SHOULD provide a way for both the NEA Client and the NEA Server to initiate a posture assessment or reassessment as needed.

PT-TCP meets this requirement. PT-TCP allows either the NEA Client or the NEA Server to initiate an assessment or reassessment.

Tunnel EAP does not meet this requirement. The typical use case scenario for using a Tunnel EAP method is to service the layer 2 network stack. In this use case, the endpoint would not have an IP address yet as it is requesting network access and thus would not be able to accept requests from the NEA Server. However, once network access has been granted, then yes, the NEA Client could receive (re)assessment requests from the NEA Server.

A.3. Evaluation Against Requirement C-3

Requirement C-3 states:

C-3 NEA protocols including security capabilities MUST be capable of protecting against active and passive attacks by intermediaries and

endpoints including prevention from replay based attacks.

PT-TCP meets this requirement. TLS includes mechanisms that provide strong cryptographic authentication, message integrity and confidentiality for NEA. In addition, to further mitigate man-in-the middle attacks, the use of channel binding at the PA layer must be used.

Tunnel EAP meets this requirement. All available Tunnel EAP methods are based on the TLS design which provide strong cryptographic authentication, message integrity and confidentiality for NEA. In addition, to further mitigate man-in-the middle attacks, the use of channel binding at the PA layer must be used.

A.4. Evaluation Against Requirement C-4

Requirement C-4 states:

C-4 The PA and PB protocols MUST be capable of operating over any PT protocol.

This requirement is not applicable to PT, though the PT-TCP protocol is independent of both the PA and PB layer.

This requirement is not applicable to PT, though the Tunnel EAP protocols are independent of both the PA and PB layer.

A.5. Evaluation Against Requirement C-5

Requirement C-5 states:

C-5 The selection process for NEA protocols MUST evaluate and prefer the reuse of existing open standards that meet the requirements before defining new ones. The goal of NEA is not to create additional alternative protocols where acceptable solutions already exist.

As TLS is a widely used open standard, it should meet this requirement.

As EMU is still in the early stages of standardizing a Tunnel EAP method, this specification reuses already widely deployed, published Tunnel EAP methods. Rather than defining a new Tunnel EAP method, this specification proposes to adopt already used ones and provides guidance for how new Tunnel EAP methods can meet this criteria to allow for NEA to use the method standardized by EMU at some future date.

A.6. Evaluation Against Requirement C-6

Requirement C-6 states:

C-6 NEA protocols MUST be highly scalable; the protocols MUST support many Posture Collectors on a large number of NEA Clients to be assessed by numerous Posture Validators residing on multiple NEA Servers.

PT-TCP meets this requirement. As PT-TCP is a protocol to establish a protected channel by which NEA data can be transported, it is independent of the content of the data it is transporting and thus can allow for carrying batches of data to multiple Posture Validators or Posture Collectors.

Tunnel EAP methods meet this requirement. As the Tunnel EAP methods define a protected transport channel that is independent of the content it transports, it can carry batches of data from and to multiple Posture Collectors and Posture Validators.

A.7. Evaluation Against Requirement C-7

Requirement C-7 states:

C-7 The protocols MUST support efficient transport of a large number of attribute messages between the NEA Client and the NEA Server.

PT-TCP meets this requirement. The PT-TCP usurps 6 octets of overhead per PT-TCP message; a small overhead to the ability to carry very many PA-TNC attributes within a PB-TNC batch.

The Tunnel EAP methods meet this requirements subject to the limitations of the underlying EAP protocol and encapsulation mechanisms. Note that a typical use case for the Tunnel EAP methods is that the assessments are brief and used for enabling network access; as such, it is not recommended to use Tunnel EAP methods to carry large amounts of attributes.

A.8. Evaluation Against Requirement C-8

Requirement C-8 states:

C-8 NEA protocols MUST operate efficiently over low bandwidth or high latency links.

PT-TCP meets this requirement. As TLS was originally designed to work at the TCP layer, it has been proven to work well over either low bandwidth or high latency links.

EAP Tunnel methods meet this requirement. The underlying EAP framework was designed and proven to work under constrained and low latency links.

A.9. Evaluation Against Requirement C-9

Requirement C-9 states:

C-9 For any strings intended for display to a user, the protocols MUST support adapting these strings to the user's language preferences.

PT-TCP meets this requirement. The PT-TCP protocol does not define messages intended for display to the user.

EAP Tunnel methods meet this requirement. The EAP Tunnel methods do not define messages intended for display to the user.

A.10. Evaluation Against Requirement C-10

Requirement C-10 states:

C-10 NEA protocols MUST support encoding of strings in UTF-8 format.

PT-TCP meets this requirement. The PT-TCP protocol does not define messages intended for display to the user.

EAP Tunnel methods meet this requirement. The EAP Tunnel methods do not define messages intended for display to the user.

A.11. Evaluation Against Requirement C-11

Requirement C-11 states:

C-11 Due to the potentially different transport characteristics provided by the underlying candidate PT protocols, the NEA Client and the NEA Server MUST be capable of becoming aware of and adapting to the limitations of the available PT protocol.

PT-TCP meets this requirement. The PT-TCP protocol uses TLS which is designed to provide reliable transport that can adapt to constrained or low bandwidth links.

EAP Tunnel methods meet this requirement. The EAP Tunnel methods are based on TLS which is designed to provide reliable transport and have been proven to adapt and work well under high latency or low bandwidth conditions.

A.12. Evaluation Against Requirement PT-1

Requirement PT-1 states:

PT-1 The PT protocol MUST NOT interpret the contents of PB messages being transported, i.e., the data it is carrying must be opaque to it.

PT-TCP meets this requirement. The PT-TCP protocol encapsulates PB messages in a TLV container without interpreting their contents.

EAP Tunnel methods meet this requirement. The EAP Tunnel methods define encapsulations for carrying arbitrary data without interpreting their contents.

A.13. Evaluation Against Requirement PT-2

Requirement PT-2 states:

PT-2 The PT protocol MUST be capable of supporting mutual authentication, integrity, confidentiality, and replay protection of the PB messages between the Posture Transport Client and the Posture Transport Server.

PT-TCP meets this requirement. The PT-TCP protocol uses TLS to provide mutual authentication, integrity, confidentiality, and replay protection.

EAP Tunnel methods meet this requirement. The EAP Tunnel methods are based on TLS which is designed to provide mutual authentication, integrity, confidentiality, and replay protection.

A.14. Evaluation Against Requirement PT-3

Requirement PT-3 states:

PT-3 The PT protocol MUST provide reliable delivery for the PB protocol. This includes the ability to perform fragmentation and reassembly, detect duplicates, and reorder to provide in-sequence delivery, as required.

PT-TCP meets this requirement. The PT-TCP protocol is designed to work over TCP which provides the fragmentation and reassembly services, detect duplicates and reorder messages if they arrive out of order.

EAP Tunnel methods meet this requirement. The EAP Tunnel methods are based on the EAP framework which provides retransmissions, while

reordering and fragmentation are handled by the individual EAP Tunnel methods.

A.15. Evaluation Against Requirement PT-4

Requirement PT-4 states:

PT-4 The PT protocol SHOULD be able to run over existing network access protocols such as 802.1X and IKEv2.

PT-TCP does NOT meet this requirement as it is designed to operate over TCP.

EAP Tunnel methods meet this requirement. The EAP Tunnel methods are based on EAP which has been enabled on both 802.1X and IKEv2.

A.16. Evaluation Against Requirement PT-5

Requirement PT-5 states:

PT-5 The PT protocol SHOULD be able to run between a NEA Client and NEA Server over TCP or UDP (similar to Lightweight Directory Access Protocol (LDAP))

PT-TCP meets this requirement. The PT-TCP protocol is designed to operate over a TCP connection.

EAP Tunnel methods do NOT meet this requirement. The EAP Tunnel methods are designed to work pre-network admission and thus are not able to communicate at the IP layer.

A.17. Evaluation Against Requirement PT-6

Requirement PT-6 states:

PT-6 The PT protocol MUST be connection oriented; it MUST support confirmed initiation and close down.

PT-TCP meets this requirement. The PT-TCP protocol is designed to operate over a TCP connection which is connection oriented and supports initiation and tear down of the connection.

EAP Tunnel methods meet this requirement. The EAP Tunnel methods are based on EAP which provides both initiation and shutdown.

A.18. Evaluation Against Requirement PT-7

Requirement PT-7 states:

PT-7 The PT protocol MUST be able to carry binary data.

PT-TCP meets this requirement. The PT-TCP protocol is capable of carrying binary data.

EAP Tunnel methods meet this requirement. The EAP Tunnel methods are capable of carrying binary data.

A.19. Evaluation Against Requirement PT-8

Requirement PT-8 states:

PT-8 The PT protocol MUST provide mechanisms for flow control and congestion control.

PT-TCP meets this requirement. The PT-TCP protocol operates over TCP which provides flow control and congestion control.

EAP Tunnel methods meet this requirement. The EAP Tunnel methods are based on EAP which, by use of the half-duplex, round-robin message exchange, flow and congestion control are provided.

A.20. Evaluation Against Requirement PT-9

Requirement PT-9 states:

PT-9 The PT protocol specifications MUST describe the capabilities that they provide for and limitations that they impose on the PB protocol (e.g. half/full duplex, maximum message size).

PT-TCP meets this requirement. This specification has provided the required information.

EAP Tunnel methods meet this requirement. This specification has provided the required information.

Authors' Addresses

Nancy Cam-Winget
Cisco Systems
80 West Tasman Drive
San Jose, CA 95134
US

Email: ncamwing@cisco.com

Joseph Salowey
Cisco Systems
2901 Third Avenue
Seattle, WA 98121
US

Email: jsalowey@cisco.com

Hao Zhou
Cisco Systems
4125 Highlander Parkway
Richfield, OH 44286
US

Email: hzhou@cisco.com

EMU Working Group
Internet-Draft
Intended status: Standards Track
Expires: November 25, 2012

S. Hartman, Ed.
Painless Security
T. Clancy
Department of Electrical
Engineering and Computer Science
K. Hooper
Motorola Solutions, Inc.
May 24, 2012

Channel Binding Support for EAP Methods
draft-ietf-emu-chbind-16.txt

Abstract

This document defines how to implement channel bindings for Extensible Authentication Protocol (EAP) methods to address the lying Network Access Service (NAS) as well as the lying provider problem.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

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."

This Internet-Draft will expire on November 25, 2012.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Introduction	5
2. Terminology	6
3. Problem Statement	6
4. Channel Bindings	8
4.1. Types of EAP Channel Bindings	8
4.2. Channel Bindings in the Secure Association Protocol	10
4.3. Channel Bindings Scope	11
5. Channel Binding Process	13
5.1. Protocol Operation	13
5.2. Channel Binding Consistency Check	15
5.3. EAP Protocol	16
5.3.1. Channel Binding Codes	18
5.3.2. Namespace Identifiers	18
5.3.3. RADIUS Namespace	18
6. System Requirements	19
6.1. General Transport Protocol Requirements	19
6.2. EAP Method Requirements	20
7. Channel Binding TLV	20
7.1. Requirements for Lower-Layer Bindings	20
7.2. EAP Lower Layer Attribute	21
8. AAA-Layer Bindings	21
9. Security Considerations	22
9.1. Trust Model	22
9.2. Consequences of Trust Violation	24
9.3. Bid-Down Attacks	25
9.4. Privacy Violations	25
10. Operations and Management Considerations	26
11. IANA Considerations	26
11.1. EAP Lower Layers Registry	27
11.2. RADIUS Registration	27
12. Acknowledgments	27
13. References	28
13.1. Normative References	28
13.2. Informative References	28

Appendix A. Attacks Prevented by Channel Bindings	29
A.1. Enterprise Subnetwork Masquerading	30
A.2. Forced Roaming	30
A.3. Downgrading attacks	30
A.4. Bogus Beacons in IEEE 802.11r	31
A.5. Forcing false authorization in IEEE 802.11i	31
Appendix B. Change History	31
B.1. Changes Since 09	31
B.2. Changes since Version 06	32
B.3. Changes since version 04	32
Authors' Addresses	32

1. Introduction

The so-called "lying NAS" problem is a well-documented problem with the current Extensible Authentication Protocol (EAP) architecture [RFC3748] when used in pass-through authenticator mode. Here, a Network Access Server (NAS), or pass-through authenticator, may represent one set of information (e.g. network identity, capabilities, configuration, etc) to the backend Authentication, Authorization, and Accounting (AAA) infrastructure, while representing contrary information to EAP peers. Another possibility is that the same false information could be provided to both the EAP peer and EAP server by the NAS. A "lying" entity can also be located anywhere on the AAA path between the NAS and the EAP server.

This problem results when the same credentials are used to access multiple services that differ in some interesting property. The EAP server learns which client credentials are in use. The client knows which EAP credentials are used, but cannot distinguish between servers that use those credentials. For methods that distinguish between client and server credentials, either using different server credentials for access to the different services or having client credentials with access to a disjoint set of services can potentially defend against the attack.

As a concrete example, consider an organization with two different IEEE 802.11 wireless networks. One is a relatively low-security network for accessing the web while the other has access to valuable confidential information. An access point on the web network could act as a lying NAS, sending the SSID of the confidential network in its beacons. This access point could gain an advantage by doing so if it tricks clients intending to connect to the confidential network to connect to it and disclose confidential information.

A similar problem can be observed in the context of roaming. Here, the lying entity is located in a visited service provider network, e.g. attempting to lure peers to connect to the network based on false advertized roaming rates. This is referred to as "the lying provider" problem in the remainder of this document. The lying entity's motivation often is financial; the entity may be paid whenever peers roam to its service. However a lying entity in a provider network can also gain access to traffic that it might not otherwise see.

This document defines and implements EAP channel bindings to solve the lying NAS and the lying provider problems, using a process in which the EAP peer provides information about the characteristics of the service provided by the authenticator to the AAA server protected within the EAP method. This allows the server to verify the

authenticator is providing information to the peer that is consistent with the information received from this authenticator as well as the information stored about this authenticator. "AAA Payloads" defined in [I-D.clancy-emu-aaapay] served as the starting point for the mechanism proposed in this specification to carry this information.

2. Terminology

In this document, several words are used to signify the requirements of the specification. These words are often capitalized. 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].

3. Problem Statement

In a [RFC4017] compliant EAP authentication, the EAP peer and EAP server mutually authenticate each other, and derive keying material. However, when operating in pass-through mode, the EAP server can be far removed from the authenticator both in terms of network distance and number of entities who need to be trusted in order to establish trusted communication. A malicious or compromised authenticator may represent incorrect information about the network to the peer in an effort to affect its operation in some way. Additionally, while an authenticator may not be compromised, other compromised elements in the network (such as proxies) could provide false information to the authenticator that it could simply be relaying to EAP peers. Hence, the goal must be to ensure that the authenticator is providing correct information to the EAP peer during the initial network discovery, selection, and authentication.

There are two different types of networks to consider: enterprise networks and service provider networks. In enterprise networks, assuming a single administrative domain, it is feasible for an EAP server to have information about all the authenticators in the network. In service provider networks, global knowledge is infeasible due to indirection via roaming. When a peer is outside its home administrative domain, the goal is to ensure that the level of service received by the peer is consistent with the contractual agreement between the two service providers. The same EAP server may need to support both types of networks. For example an enterprise may have a roaming agreement permitting its users to use the networks of third-party service providers. In these situations, the EAP server may authenticate for an enterprise and provider network.

The following are example attacks possible by presenting false

network information to peers.

- o Enterprise Network: A corporate network may have multiple virtual LANs (VLANs) available throughout their campus network, and have IEEE 802.11 access points connected to each VLAN. Assume one VLAN connects users to the firewalled corporate network, while the other connects users to a public guest network. The corporate network is assumed to be free of adversarial elements, while the guest network is assumed to possibly have malicious elements. Access Points on both VLANs are serviced by the same EAP server, but broadcast different SSIDs to differentiate. A compromised access point connected to the guest network but not the corporate network could advertise the SSID of the corporate network in an effort to lure peers to connect to a network with a false sense of security regarding their traffic. Conditions and further details of this attack can be found in the Appendix.
- o Enterprise network: The EAP GSS-API mechanism [I-D.ietf-abfab-gss-eap] mechanism provides a way to use EAP to authenticate to mail servers, instant messaging servers and other non-network services. Without EAP channel binding, an attacker could trick the user into connecting to a relatively untrusted service instead of a relatively trusted service. For example the instant messaging service could impersonate the mail server.
- o Service Provider Network: An EAP-enabled mobile phone provider could advertise very competitive flat rates but send per minute rates to the home server, thus, luring peers to connect to their network and overcharging them. In more elaborate attacks, peers can be tricked into roaming without their knowledge. For example, a mobile phone provider operating along a geo-political boundary could boost their cell towers' transmission power and advertise the network identity of the neighboring country's indigenous provider. This would cause unknowing handsets to associate with an unintended operator, and consequently be subject to high roaming fees without realizing they had roamed off their home provider's network. These types of scenarios can be considered as "lying provider" problem, because here the provider configures its NAS to broadcast false information. For the purpose of channel bindings as defined in this draft, it does not matter which local entity (or entities) is "lying" in a service provider network (local NAS, local authentication server and/or local proxies), because the only information received from the visited network that is verified by channel bindings is the information the home authentication server received from the last hop in the communication chain. In other words, channel bindings enable the detection of inconsistencies in the information from a visited network, but cannot determine which entity is lying. Naturally,

channel bindings for EAP methods can only verify the endpoints and, if desirable, intermediate hops need to be protected by the employed AAA protocol.

- o Enterprise and provider networks: In a situation where an enterprise has roaming agreements with providers, a compromised access point in a provider network could masquerade as the enterprise network in an attempt to gain confidential information. Today this could potentially be solved by using different credentials for internal and external access. Depending on the type of credential this may introduce usability or man-in-the-middle security issues.

To address these problems, a mechanism is required to validate unauthenticated information advertised by EAP authenticators.

4. Channel Bindings

EAP channel bindings seek to authenticate previously unauthenticated information provided by the authenticator to the EAP peer, by allowing the peer and server to compare their perception of network properties in a secure channel.

It should be noted that the definition of EAP channel bindings differs somewhat from channel bindings documented in [RFC5056], which seek to securely bind together the end points of a multi-layer protocol, allowing lower layers to protect data from higher layers. Unlike [RFC5056], EAP channel bindings do not ensure the binding of different layers of a session but rather the information advertised to EAP peer by an authenticator acting as pass-through device during an EAP execution. The term channel bindings was independently adopted by these two related concepts; by the time the conflict was discovered, a wide body of literature existed for each usage. EAP channel bindings could be used to provide RFC 5056 channel bindings. In particular, an inner EAP method could be bound to an outer method by including the RFC 5056 channel binding data for the outer channel in the inner EAP method's channel bindings. Doing so would provide a facility similar to EAP cryptographic binding, except that a man-in-the-middle could not extract the inner method from the tunnel. This specification does not weigh the advantages of doing so nor specify how to do so; the example is provided only to illustrate how EAP channel binding and RFC 5056 channel binding overlap.

4.1. Types of EAP Channel Bindings

There are two categories of approach to EAP channel bindings:

- o After keys have been derived during an EAP execution, the peer and server can, in an integrity-protected channel, exchange plaintext information about the network with each other, and verify consistency and correctness.
- o The peer and server can both uniquely encode their respective view of the network information without exchanging it, resulting into an opaque blob that can be included directly into the derivation of EAP session keys.

Both approaches are only applicable to key deriving EAP methods and both have advantages and disadvantages. Various hybrid approaches are also possible. Advantages of exchanging plaintext information include:

- o It allows for policy-based comparisons of network properties, rather than requiring precise matches for every field, which achieves a policy-defined consistency, rather than bitwise equality. This allows network operators to define which properties are important and even verifiable in their network.
- o EAP methods that support extensible, integrity-protected channels can easily include support for exchanging this network information. In contrast, direct inclusion into the key derivation would require more extensive revisions to existing EAP methods or a wrapper EAP method.
- o Given it doesn't affect the key derivation, this approach facilitates debugging, incremental deployment, backward compatibility and a logging mode in which verification results are recorded but do not have an effect on the remainder of the EAP execution. The exact use of the verification results can be subject to the network policy. Additionally, consistent information canonicalization and formatting for the key derivation approach would likely cause significant deployment problems.

The following are advantages of directly including channel binding information in the key derivation:

- o EAP methods not supporting extensible, integrity-protected channels could still be supported, either by revising their key derivation, revising EAP, or wrapping them in a universal method that supports channel binding.
- o It can guarantee proper channel information, since subsequent communication would be impossible if differences in channel information yielded different session keys on the EAP peer and server.

4.2. Channel Bindings in the Secure Association Protocol

This document describes channel bindings performed by transporting channel binding information as part of an integrity-protected exchange within an EAP method. Alternatively, some future document could specify a mechanism for transporting channel bindings within the lower layer's secure association protocol. Such a specification would need to describe how channel bindings are exchanged over the lower layer protocol between the peer and authenticator. In addition, since the EAP exchange concludes before the secure association protocol begins, a mechanism for transporting the channel bindings from the authenticator to the EAP server needs to be specified. A mechanism for transporting a protected result from the EAP server, through the authenticator, back to the peer needs to be specified.

The channel bindings **MUST** be transported with integrity protection based on a key known only to the peer and EAP server. The channel bindings **SHOULD** be confidentiality protected using a key known only to the peer and EAP server. For the system to function, the EAP server or AAA server needs access to the channel binding information from the peer as well as the AAA attributes and a local database described later in this document.

The primary advantage of sending channel bindings as part of the secure association protocol is that EAP methods need not be changed. The disadvantage is that a new AAA exchange is required, and secure association protocols need to be changed. As the result of the secure association protocol change, every NAS needs to be upgraded to support channel bindings within the secure association protocol.

For many deployments, changing all the NASes is expensive and adding channel binding support to enough EAP methods to meet the goals of the deployment will be cheaper. However for deployment of new equipment, or especially deployment of a new lower layer technology, changing the NASes may be cheaper than changing EAP methods. Especially if such a deployment needed to support a large number of EAP methods, sending channel bindings in the secure association protocol might make sense. Sending channel bindings in the secure association protocol can work even with ERP [RFC5296] in which previously established EAP key material is used for the secure association protocol without carrying out any EAP method during re-authentication.

If channel bindings using a secure association protocol is specified, semantics as well as the set of information that peers exchange can be shared with the mechanism described in this document.

4.3. Channel Bindings Scope

The scope of EAP channel bindings differs somewhat depending on the type of deployment in which they are being used. In enterprise networks, they can be used to authenticate very specific properties of the authenticator (e.g. MAC address, supported link types and data rates, etc), while in service provider networks they can generally only authenticate broader information about a roaming partner's network (e.g. network name, roaming information, link security requirements, etc). The reason for the difference has to do with the amount of information about the authenticator and/or network to which the peer is connected the home EAP server is expected to have access to. In roaming cases, the home server is likely to only have access to information contained in their roaming agreements.

With any multi-hop AAA infrastructure, many of the NAS-specific AAA attributes are obscured by the AAA proxy that's decrypting, reframing, and retransmitting the underlying AAA messages. Especially service provider networks are affected by this and the AAA information received from the last hop may not contain much verifiable information any longer. For example, information carried in AAA attributes such as the NAS IP address may have been lost in transition and are thus not known to the EAP server. Even worse, information may still be available but be useless, for example representing the identity of a device on a private network or a middlebox. This affects the ability of the EAP server to verify specific NAS properties. However, often verification of the MAC or IP address of the NAS is not useful for improving the overall security posture of a network. More often the best approach is to make policy decisions about services being offered to peers. For example, in an IEEE 802.11 network, the EAP server may wish to ensure that peers connecting to the corporate intranet are using secure link-layer encryption, while link-layer security requirements for peers connecting to the guest network could be less stringent. These types of policy decisions can be made without knowing or being able to verify the IP address of the NAS through which the peer is connecting.

The properties of the network that the peer wishes to validate depend on the specific deployment. In a mobile phone network, peers generally don't care what the name of the network is, as long as they can make their phone call and are charged the expected amount for the call. However, in an enterprise network the administrators of a peer may be more concerned with specifics of where their network traffic is being routed and what VLAN is in use. To establish policies surrounding these requirements administrators would capture some attribute such as SSID to describe the properties of the network they care about. Channel bindings could validate the SSID. The

administrator would need to make sure that the network guarantees that when an authenticator trusted by the AAA infrastructure to offer a particular SSID to clients does offer this SSID, that network has the intended properties. Generally it is not possible for channel bindings to detect lying NAS behavior when the NAS is authorized to claim a particular service. That is, if the same physical authenticator is permitted to advertise two networks, the AAA infrastructure is unlikely to be able to determine when this authenticator lies.

As discussed in the next section, some of the most important information to verify cannot come from AAA attributes but instead comes from local configuration. For example in the mobile phone case, the expected roaming rate cannot come from the roaming provider without being verified against the contract between the two providers. Similarly, in an enterprise, the SSID a particular access point is expected to advertise is a matter of configuration rather than something that can be trusted because it is included in an AAA exchange.

The peer and authenticator do not initially have a basis for trust. The peer has a credential with the EAP server that forms a basis for trust. The EAP server and authenticator have a potentially indirect trust path using the AAA infrastructure. Channel binding leverages the trust between the peer and EAP server to build trust in certain attributes between the peer and authenticator.

Channel bindings can be important for forming areas of trust, especially when provider networks are involved, and exact information is not available to the EAP server. Without channel bindings, all entities in the system need to be held to the standards of the most trusted entity that could be accessed using the EAP credential. Otherwise, a less trusted entity can impersonate a more trusted entity. However when channel bindings are used, the EAP server can use information supplied by the peer, AAA protocols and local database to distinguish less trusted entities from more trusted entities. One possible deployment involves being able to verify a number of characteristics about relatively trusted entities while for other entities simply verifying that they are less trusted.

Any deployment of channel bindings should take into consideration both what information the EAP server is likely to know or have access to, and also what type of network information the peer would want and need authenticated.

5. Channel Binding Process

This section defines the process for verifying channel binding information during an EAP authentication. The protocol uses the approach where plaintext data is exchanged, since it allows channel bindings to be used more flexibly in varied deployment models (see Section 4.1). In the first subsection, the general communication infrastructure is outlined, the messages used for channel binding verifications are specified, and the protocol flows are defined. The second subsection explores the difficulties of checking the different pieces of information that are exchanged during the channel binding protocol for consistency. The third subsection describes the information carried in the EAP exchange.

5.1. Protocol Operation

Channel bindings are always provided between two communication endpoints, here the EAP peer and the EAP server, who communicate through an authenticator typically in pass-through mode. Specifications treat the AAA server and EAP server as distinct entities. However there is no standardized protocol for the AAA server and EAP server to communicate with each other. For the channel binding protocol presented in this draft to work, the EAP server needs to be able to access information from the AAA server that is utilized during the EAP session (i2 below) and a local database. For example, the EAP server and the local database can be co-located with the AAA server, as illustrated in Figure 1. An alternate architecture would be to provide a mechanism for the EAP server to inform the AAA server what channel binding attributes were supplied and the AAA server to inform the EAP server about what channel binding attributes it considered when making its decision.

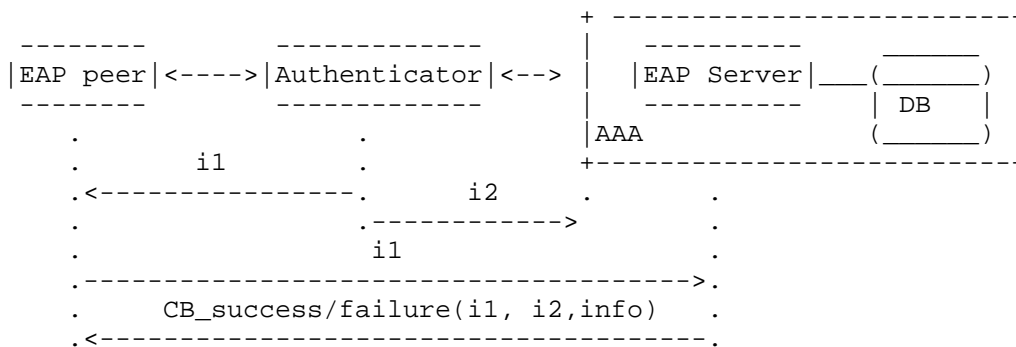


Figure 1: Overview of Channel Binding Protocol

During network advertisement, selection, and authentication, the authenticator presents unauthenticated information, labeled i1, about the network to the peer. Message i1 could include an authenticator identifier and the identity of the network it represents, in addition to advertised network information such as offered services and roaming information. Information may be communicated implicitly in i1, such as the type of media in use. As there is no established trust relationship between the peer and authenticator, there is no way for the peer to validate this information.

Additionally, during the transaction the authenticator presents a number of information properties in form of AAA attributes about itself and the current request to the AAA infrastructure which may or may not be valid. This information is labeled i2. Message i2 is the information the AAA server receives from the last hop in the AAA proxy chain which is not necessarily the authenticator.

AAA hops between the authenticator and AAA server can validate some of i2. Whether the AAA server will be able to depend on this depends significantly on the business relationship executed with these proxies and on the structure of the AAA network.

The local database is perhaps the most important part of this system. In order for the EAP server or AAA server to know whether i1 and i2 are correct, they need access to trustworthy information, since an authenticator could include false information in both i1 and i2. Additional reasons why such a database is necessary for channel bindings to work are discussed in the next subsection. The information contained within the database could involve wildcards. For example, this could be used to check whether WiFi access points on a particular IP subnet all use a specific SSID. The exact IP address is immaterial, provided it is on the correct subnet.

During an EAP method execution with channel bindings, the peer sends i1 to the EAP server using the mechanism described in Section 5.3. the EAP server verifies the consistency of i1 provided by the peer, i2 provided by the authenticator, and the information in the local database. Upon the check, the EAP server sends a message to the peer indicating whether the channel binding validation check succeeded or failed and includes the attributes that were used in the check. The message flow is illustrated in Figure 1.

Above, the EAP server is described as performing the channel binding validation. In most deployments, this will be a necessary implementation constraint. The EAP exchange needs to include an indication of channel binding success or failure. Most existing implementations do not have a way to have an exchange between the EAP server and another AAA entity during the EAP server's processing of a

single EAP message. However another AAA entity can provide information to the EAP server to make its decision.

If the compliance of i1 or i2 information with the authoritative policy source is mandatory and a consistency check failed, then after sending a protected indication of failed consistency, the EAP server MUST send an EAP-Failure message to terminate the session. If the EAP server is otherwise configured, it MUST allow the EAP session to complete normally, and leave the decision about network access up to the peer's policy. If i1 or i2 does not comply with policy, the EAP server MUST NOT list information that failed to comply in the set of information used to perform channel binding. In this case the EAP server SHOULD indicate channel binding failure; this requirement may be upgraded to a MUST in the future.

5.2. Channel Binding Consistency Check

The validation check that is the core of the channel binding protocol described in the previous subsection, consists of two parts in which the server checks whether:

1. the authenticator is lying to the peer, i.e. i1 contains false information,
2. the authenticator or any entity on the AAA path to the AAA server provides false information in form of AAA attributes, i.e. i2 contains false information,

These checks enable the EAP server to detect lying NAS/authenticator in enterprise networks and lying providers in service provider networks.

Checking the consistency of i1 and i2 is nontrivial, as has been pointed out already in [HC07]. First, i1 can contain any type of information propagated by the authenticator, whereas i2 is restricted to information that can be carried in AAA attributes. Second, because the authenticator typically communicates over different link layers with the peer and the AAA infrastructure, different type of identifiers and addresses may have been presented to both communication endpoints. Whether these different identifiers and addresses belong to the same device cannot be directly checked by the EAP server or AAA server without additional information. Finally, i2 may be different from the original information sent by the authenticator because of en route processing or malicious modifications. As a result, in the service provider model, typically the i1 information available to the EAP server can only be verified against the last-hop portion of i2, or values propagated by proxy servers. In addition, checking the consistency of i1 and i2 alone is

insufficient because an authenticator could lie to both, the peer and the EAP server, i.e. i1 and i2 may be consistent but both contain false information.

A local database is required to leverage the above mentioned shortcomings and support the consistency and validation checks. In particular, information stored for each NAS/authenticator (enterprise scenario) or each roaming partner (service provider scenario) enables a comparison of any information received in i1 with AAA attributes in i2 as well as additionally stored AAA attributes that might have been lost in transition. Furthermore, only such a database enables the EAP server and AAA server to check the received information against trusted information about the network including roaming agreements.

Section 7 describes lower-layer specific properties that can be exchanged as a part of i1. Section 8 describes specific AAA attributes that can be included and evaluated in i2. The EAP server reports back the results from the channel binding validation check that compares the consistency of all the values with those in the local database. The challenges of setting up such a local database are discussed in Section 10.

5.3. EAP Protocol

EAP methods supporting channel binding consistent with this specification provide a mechanism for carrying channel binding data from the peer to the EAP server and a channel binding response from the EAP server to the peer. The specifics of this mechanism are dependent on the method, although the content of the channel binding data and channel binding response are defined by this section.

Typically the lower layer will communicate a set of attributes to the EAP implementation on the peer that should be part of channel binding. The EAP implementation may need to indicate to the lower layer that channel binding information cannot be sent. Reasons for failing to send channel binding information include an EAP method that does not support channel binding is selected, or channel binding data is too big for the EAP method selected. Peers SHOULD provide appropriate policy controls to select channel binding or mandate its success.

The EAP server receives the channel binding data and performs the validation. The EAP method provides a way to return a response; the channel binding response uses the same basic format as the channel binding data.

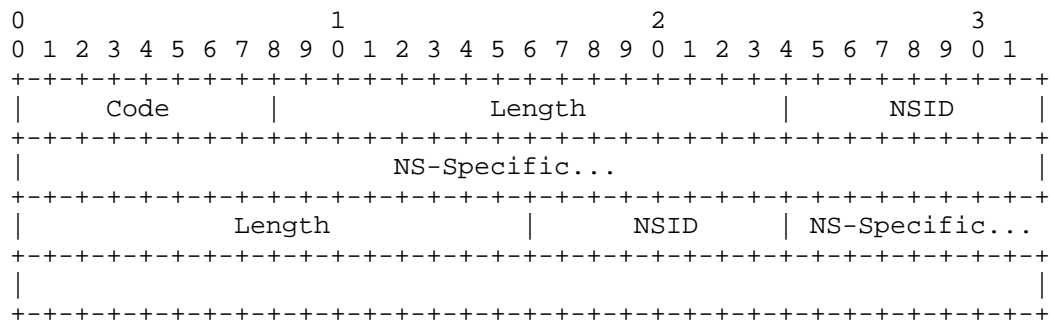


Figure 2: Channel Binding Encoding

Both the channel binding data and response use the format illustrated in Figure 2. The protocol starts with a one byte code; see Section 5.3.1. Then for each type of attributes contained in the channel binding data, the following information is encoded:

Length: Two octets of length in network byte order, indicating the length of the NS-SPECIFIC data. The NSID and length octets are not included.

NSID: One octet describing the namespace from which the attributes are drawn. See Section 5.3.3 for a description of how to encode RADIUS attributes in channel binding data and responses. RADIUS uses a namespace identifier of 1.

NS-SPECIFIC: The encoding of the attributes in a manner specific to the type of attribute.

A given NSID MUST NOT appear more than once in a channel binding data or channel binding response. Instead, all NS-SPECIFIC data for a particular NSID must occur inside one (NSID, Length, NS-Specific) field. This set of fields may be repeated if multiple namespaces are included.

In channel binding data, the code is set to 1 (channel binding data) and the full attributes and values that the peer wishes the EAP server to validate are included.

In a channel binding response, the server selects the code; see Section 5.3.1. For successful channel binding, the server returns code 2. The set of attributes that the EAP server returns depend on the code. For success, the server returns the attributes that were considered by the server in making the determination that channel bindings are successfully validated; attributes that the server is

unable to check or that failed to validate against what is sent by the peer MUST NOT be returned in a success response. Generally, servers will not return a success response if any attributes were checked and failed to validate those specified by the peer. Special circumstances such as a new attribute being phased in at a server MAY require servers to return success when such an attribute fails to validate. The server returns the value supplied by the peer when returning an attribute in channel binding responses.

For channel binding failure (code 3), the server SHOULD include any attributes that were successfully validated. This code means that server policy indicates that the attributes sent by the client do not accurately describe the authenticator. Servers MAY include no attributes in this response, for example if all the attributes supplied by the peer that the server can check failed to be consistent.

Peers MUST treat unknown codes as Channel binding Failure. Peers MUST ignore differences between attribute values sent in the channel binding data and those sent in the response. Peers and servers MUST ignore any attributes contained in a field with an unknown NSID. Peers MUST ignore any attributes in a response not present in the channel binding data.

5.3.1. Channel Binding Codes

Code	Meaning
1	Channel Binding data from client
2	Channel binding response: success
3	Channel binding response: failure

5.3.2. Namespace Identifiers

ID	Namespace	Reference
1	RADIUS	Section 5.3.3
255	Private Use	

5.3.3. RADIUS Namespace

RADIUS AVPs are encoded with a one-octet attribute type followed by a one-octet length followed by the value of the RADIUS attribute being encoded. The length includes the type and length octets; the minimum

legal length is 3. Attributes are concatenated to form the namespace specific portion of the packet.

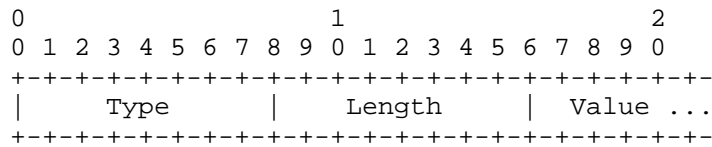


Figure 3: RADIUS AVP Encoding

The full value of an attribute is included in the channel binding data and response.

6. System Requirements

This section defines requirements on components used to implement the channel bindings protocol.

The channel binding protocol defined in this document must be transported after keying material has been derived between the EAP peer and server, and before the peer would suffer adverse affects from joining an adversarial network. This document describes a protocol for performing channel binding within EAP methods. As discussed in Section 4.2, an alternative approach for meeting this requirement is to perform channel bindings during the secure association protocol of the lower layer.

6.1. General Transport Protocol Requirements

The transport protocol for carrying channel binding information MUST support end-to-end (i.e. between the EAP peer and server) message integrity protection to prevent the adversarial NAS or AAA device from manipulating the transported data. The transport protocol SHOULD provide confidentiality. The motivation for this is that the channel bindings could contain private information, including peer identities, which SHOULD be protected. If confidentiality cannot be provided, private information MUST NOT be sent as part of the channel binding information.

Any transport needs to be careful not to exceed the MTU for its lower-layer medium. In particular, if channel binding information is exchanged within protected EAP method channels, these methods may or may not support fragmentation. In order to work with all methods, the channel binding messages must fit within the available payload. For example, if the EAP MTU is 1020 octets, and EAP-GPSK is used as

the authentication method, and maximal-length identities are used, a maximum of 384 octets are available for conveying channel binding information. Other methods, such as EAP-TTLS, support fragmentation and could carry significantly longer payloads.

6.2. EAP Method Requirements

If transporting data directly within an EAP method, it MUST be able to carry integrity protected data from the EAP peer to server. EAP methods SHOULD provide a mechanism to carry protected data from server to peer. EAP methods MUST exchange channel binding data with the AAA subsystem hosting the EAP server. EAP methods MUST be able to import channel binding data from the lower layer on the EAP peer.

7. Channel Binding TLV

This section defines some channel binding TLVs. While message `il` is not limited to AAA attributes, for the sake of tangible attributes that are already in place, this section discusses AAA AVPs that are appropriate for carrying channel bindings (i.e. data from `il` in Section 5).

For any lower-layer protocol, network information of interest to the peer and server can be encapsulated in AVPs or other defined payload containers. The appropriate AVPs depend on the lower layer protocol as well as on the network type (i.e. enterprise network or service provider network) and its application.

7.1. Requirements for Lower-Layer Bindings

Lower-layer protocols MUST support EAP in order to support EAP channel bindings. These lower layers MUST support EAP methods that derive keying material, as otherwise no integrity-protected channel would be available to execute the channel bindings protocol. Lower-layer protocols need not support traffic encryption, since this is independent of the authentication phase.

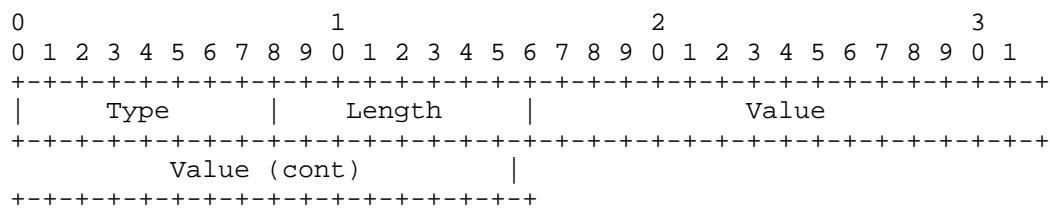
The data conveyed within the AVP type MUST NOT conflict with the externally-defined usage of the AVP. Additional TLV types MAY be defined for values that are not communicated within AAA attributes.

In general, lower layers will need to specify what information should be included in `il`. Existing lower layers will probably require new documents to specify this information. Lower layer specifications need to include sufficient information in `il` to uniquely identify which lower layer is involved. The preferred way to do this is to include the `eap-lower-layer` attribute defined in the next section.

This MUST be included in i1 unless an attribute specific to a particular lower layer is included in i1.

7.2. EAP Lower Layer Attribute

A new RADIUS attribute is defined to carry information on which EAP lower layer is used for this EAP authentication. This Attribute provides information relating to the lower layer over which EAP is transported. This Attribute MAY be sent by the NAS to the RADIUS server in an Access-Request or an Accounting-Request packet. A summary of the EAP-Lower-Layer Attribute format is shown below. The fields are transmitted from left to right.



The code is TBD, the length is 6 and the value is a 32-bit unsigned integer in network byte order. The value specifies the EAP lower layer in use. Values are taken from the IANA registry established in Section 11.1.

8. AAA-Layer Bindings

This section discusses which AAA attributes in a AAA Access-Request message can and should be validated by a EAP server (i.e. data from i2 in Section 5). As noted before, this data can be manipulated by AAA proxies either to enable functionality (e.g. removing realm information after messages have been proxied) or maliciously (e.g. in the case of a lying provider). As such, this data cannot always be easily validated. However as thorough of a validation as possible should be conducted in an effort to detect possible attacks.

NAS-IP-Address: This value is typically the IP address of the authenticator, but in a proxied connection it likely will not match the source IP address of an Access-Request. A consistency check MAY verify the subnet of the IP address was correct based on the last-hop proxy.

NAS-IPv6-Address: This value is typically the IPv6 address of the authenticator, but in a proxied connection it likely will not match the source IPv6 address of an Access-Request. A consistency check MAY verify the subnet of the IPv6 address was correct based on the last-hop proxy.

NAS-Identifier: This is an identifier populated by the NAS to identify the NAS to the AAA server; it SHOULD be validated against the local database.

NAS-Port-Type: This specifies the underlying link technology. It SHOULD be validated against the value received from the peer in the information exchange, and against a database of authorized link-layer technologies.

9. Security Considerations

This section discusses security considerations surrounding the use of EAP channel bindings.

9.1. Trust Model

In the considered trust model, EAP peer and authentication server are honest while the authenticator is maliciously sending false information to peer and/or server. In the model, the peer and server trust each other, which is not an unreasonable assumption, considering they already have a trust relationship. The following are the trust relationships:

- o The server trusts that the channel binding information received from the peer is the information that the peer received from the authenticator.
- o The peer trusts the channel binding result received from the server.
- o The server trusts the information contained within its local database.

In order to establish the first two trust relationships during an EAP execution, an EAP method MUST provide the following:

- o mutual authentication between peer and server
- o derivation of keying material including a key for integrity protection of channel binding messages known to the peer and EAP server but not the authenticator
- o sending channel binding request from peer to server over an integrity-protected channel

- o sending the channel binding result from server to peer over an integrity-protected channel

This trust model is a significant departure from the standard EAP model. In many EAP deployments today attacks where one authenticator can impersonate another are not a significant concern because all authenticators provide the same service. A authenticator does not gain significant advantage by impersonating another authenticator. The use of EAP in situations where different authenticators provide different services may give an attacker who can impersonate a authenticator greater advantage. The system as a whole needs to be analyzed to evaluate cases where one authenticator may impersonate another and to evaluate the impact of this impersonation.

One attractive implementation strategy for channel binding is to add channel binding support to a tunnel method which can tunnel an inner EAP authentication. This way, channel binding can be achieved with any method that can act as an inner method even if that inner method does not have native channel binding support. The requirement for mutual authentication and key derivation is at the layer of EAP that actually performs the channel binding. Tunnel methods sometimes use cryptographic binding, a process where a peer proves that the peer for the outer method is the same as the peer for an inner method to tie authentication at one layer together with an inner layer. Cryptographic binding does not always provide mutual authentication; its definition does not require the server to prove that the inner server and outer server are the same. Even when cryptographic binding does attempt to confirm that the inner and outer server are the same, the Master Session Key (MSK) from the inner method is typically used to protect the binding. An attacker such as an authenticator that wishes to subvert channel binding could establish an outer tunnel terminating at the authenticator. If the outer method tunnel terminates on the authenticator, the MSK is disclosed to the authenticator, which can typically attack cryptographic binding. If the authenticator controls cryptographic binding then it typically controls the channel binding parameters and results. If the channel binding process is used to differentiate one authenticator from another then the authenticator can claim to support services that it was not authorized to. This attack was not in scope for existing threat models for cryptographic binding because differentiated authenticators was not a consideration. Thus, existing cryptographic binding does not typically provide mutual authentication of the inner method server required for channel binding. Other methods besides cryptographic binding are available to provide mutual authentication required by channel binding. As an example, if server certificates are validated and names checked, mutual authentication can be provided directly by the tunnel.

9.2. Consequences of Trust Violation

If any of the trust relationships listed in Section 9.1 are violated, channel binding cannot be provided. In other words, if mutual authentication with key establishment as part of the EAP method as well as protected database access are not provided, then achieving channel binding is not feasible.

Dishonest peers can only manipulate the first message *i1* of the channel binding protocol. In this scenario, a peer sends *i1'* to the server. If *i1'* is invalid, the channel binding validation will fail. On the other hand if *i1'* passes the validation, either the original *i1* was wrong and *i1'* corrected the problem or both *i1* and *i1'* constitute valid information. A peer could potentially gain an advantage in auditing or charging if both are valid and information from *i1'* is used for auditing or charging. Such peers can be detected by including the information in *i2* and checking *i1* against *i2*.

If information from *i1* does not validate, an EAP server cannot generally determine whether the authenticator advertised incorrect information or whether the peer is dishonest. This should be considered before using channel binding validation failures to determine the reputation either of the peer or authenticator.

Dishonest servers can send EAP-Failure messages and abort the EAP authentication even if the received *i1* is valid. However, servers can always abort any EAP session independent of whether channel binding is offered or not. On the other hand, dishonest servers can claim a successful validation even if *i1* contains invalid information. This can be seen as collaboration of authenticator and server. Channel binding can neither prevent nor detect such attacks. In general such attacks cannot be prevented by cryptographic means and should be addressed using policies making servers liable for their provided information and services.

Additional network entities (such as proxies) might be on the communication path between peer and server and may attempt to manipulate the channel binding protocol. If these entities do not possess the keying material used for integrity protection of the channel binding messages, the same threat analysis applies as for the dishonest authenticators. Hence, such entities can neither manipulate single channel binding messages nor the outcome. On the other hand, entities with access to the keying material must be treated like a server in a threat analysis. Hence such entities are able to manipulate the channel binding protocol without being detected. However, the required knowledge of keying material is unlikely since channel binding is executed before the EAP method is

completed, and thus before keying material is typically transported to other entities.

9.3. Bid-Down Attacks

EAP methods that add channel binding will typically negotiate its use. Even for entirely new EAP methods designed with channel binding from the first version, some deployments may not use it. It is desirable to protect against attacks on the negotiation of channel bindings. An attacker including the NAS SHOULD NOT be able to prevent a peer and server who support channel bindings from using it.

Unfortunately existing EAP methods may make it difficult or impossible to protect against attacks on negotiation. For example, many EAP state machines will accept a success message at any point after key derivation to terminate authentication. EAP success methods are not integrity protected; an attacker who could insert a message can generate one. The NAS is always in a position to generate a success message. Common EAP servers take advantage of state machines accepting success messages even in cases where an EAP method might support a protected indication of success. It may be challenging to define channel binding support for existing EAP methods in a manner that permits peers to distinguish an old EAP server that sends a success indication and does not support channel binding from an attacker injecting a success indication.

9.4. Privacy Violations

While the channel binding information exchanged between EAP peer and EAP server (i.e. i1 and the result message) must always be integrity-protected it may not be encrypted. In the case that these messages contain identifiers of peer and/or network entities, the privacy property of the executed EAP method may be violated. Hence, in order to maintain the privacy of an EAP method, the exchanged channel binding information must be encrypted. If encryption is not available, private information is not sent as part of the channel binding information, as described in Section 6.1.

Privacy implications of attributes selected for channel binding need to be considered. Consider channel binding the username attribute. A peer sends a privacy protecting anonymous identifier in its EAP identity message, but sends the full username in the protected i1 message. However the authenticator would like to learn the full username. It makes a guess and sends that in i2 rather than the anonymous identifier. If the EAP server validates this attribute and fails when the username from the peer mismatches i2, then the EAP server confirms the authenticator's guess. Similar privacy exposures may result whenever one party is in a position to guess channel

binding information provided by another party.

10. Operations and Management Considerations

As with any extension to existing protocols, there will be an impact on existing systems. Typically the goal is to develop an extension that minimizes the impact on both development and deployment of the new system, subject to the system requirements. This section discusses the impact on existing devices that currently utilize EAP, assuming the channel binding information is transported within the EAP method execution.

The EAP peer will need an API between the EAP lower layer and the EAP method that exposes the necessary information from the NAS to be validated to the EAP peer, which can then feed that information into the EAP methods for transport. For example, an IEEE 802.11 system would need to make available the various information elements that require validation to the EAP peer which would properly format them and pass them to the EAP method. Additionally, the EAP peer will require updated EAP methods that support transporting channel binding information. While most method documents are written modularly to allow incorporating arbitrary protected information, implementations of those methods would need to be revised to support these extensions. Driver updates are also required so methods can access the required information.

No changes to the pass-through authenticator would be required.

The EAP server would need an API between the database storing NAS information and the individual EAP server. The database may already exist on the AAA server in which case the EAP server passes the parameters to the AAA server for validation. The EAP methods need to be able to export received channel binding information to the EAP server so it can be validated.

11. IANA Considerations

A new top level registry is created for "EAP Channel Binding Parameters." This registry consists of several sub registries.

The "Channel Binding Codes" sub-registry defines values for the code field in the channel binding data and channel binding response packet. See the table in Section 5.3.1 for initial registrations. This registry requires standards action [RFC5226] for new registrations. Early allocation [RFC4020] is allowed. An additional reference column should be added to the table for the registry,

pointing all codes in the initial registration to this specification. Valid values in this sub-registry range from 0-255; 0 is reserved.

The "Channel Binding Namespaces" sub-registry contains registrations for the NSID field in the channel binding data and channel binding response. Initial registrations are found in the table in Section 5.3.2. Registrations in this registry require IETF review. Valid values range from 0-255; 0 is reserved. As with the Channel Binding Codes sub-registry a reference column should be included and should point to this document for initial registrations.

11.1. EAP Lower Layers Registry

A new sub registry in the EAP Numbers registry at <http://www.iana.org/assignments/eap-numbers> is created for EAP Lower Layers. Registration requires expert review; the primary role of the expert is to prevent multiple registrations for the same lower layer.

The following table gives the initial registrations for this registry.

Value	Lower Layer
1	Wired IEEE 802.1X
2	IEEE 802.11 (no-pre-auth)
3	IEEE 802.11 (pre-authentication)
4	IEEE 802.16e
5	IKEv2
6	PPP
7	PANA (no pre-authentication) [RFC5191]
8	GSS-API [I-D.ietf-abfab-gss-eap]
9	PANA (pre-authentication) [RFC5873]

11.2. RADIUS Registration

A new RADIUS attribute is registered with the name EAP-Lower-Layer; TBD should be replaced with the number corresponding to this attribute. The RADIUS attributes are in the registry at <http://www.iana.org/assignments/radius-types/radius-types.xml>

12. Acknowledgments

The authors and editor would like to thank Bernard Aboba, Glen Zorn, Joe Salowey, Stephen Hanna, and Klaas Wierenga for their valuable inputs that helped to improve and shape this document over the time.

Sam Hartman's work on this specification is funded by JANET(UK).

The EAP-Lower-Layer attribute was taken from draft-aboba-radext-wlan [I-D.aboba-radext-wlan].

13. References

13.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowitz, "Extensible Authentication Protocol (EAP)", RFC 3748, June 2004.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.
- [RFC4020] Kompella, K. and A. Zinin, "Early IANA Allocation of Standards Track Code Points", BCP 100, RFC 4020, February 2005.

13.2. Informative References

- [I-D.aboba-radext-wlan] Aboba, B., Malinen, J., Congdon, P., and J. Salowey, "RADIUS Attributes for IEEE 802 Networks", draft-aboba-radext-wlan-15 (work in progress), October 2011.
- [I-D.clancy-emu-aaapay] Clancy, T., Lior, A., and G. Zorn, Ed., "EAP Method Support for Transporting AAA Payloads", Internet Draft draft-clancy-emu-aaapay-02, May 2009.
- [RFC4017] Stanley, D., Walker, J., and B. Aboba, "Extensible Authentication Protocol (EAP) Method Requirements for Wireless LANs", RFC 4017, March 2005.
- [RFC5056] Williams, N., "On the Use of Channel Bindings to Secure Channels", RFC 5056, November 2007.
- [HC07] Hoeper, K. and L. Chen, "Where EAP Security Claims Fail", ICST QShine, August 2007.

[80211U-D4.01]

"Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications - Amendment 7: Interworking with External Networks", IEEE Draft Standard 802.11u, November 2008.

[I-D.ietf-abfab-gss-eap]

Hartman, S. and J. Howlett, "A GSS-API Mechanism for the Extensible Authentication Protocol", draft-ietf-abfab-gss-eap-06 (work in progress), April 2012.

[RFC5873] Ohba, Y. and A. Yegin, "Pre-Authentication Support for the Protocol for Carrying Authentication for Network Access (PANA)", RFC 5873, May 2010.

[RFC5296] Narayanan, V. and L. Dondeti, "EAP Extensions for EAP Re-authentication Protocol (ERP)", RFC 5296, August 2008.

[RFC5191] Forsberg, D., Ohba, Y., Patil, B., Tschofenig, H., and A. Yegin, "Protocol for Carrying Authentication for Network Access (PANA)", RFC 5191, May 2008.

Appendix A. Attacks Prevented by Channel Bindings

In the following it is demonstrated how the presented channel bindings can prevent attacks by malicious authenticators (representing the lying NAS problem) as well as malicious visited networks (representing the lying provider problem). This document only provides part of the solution necessary to realize a defense against these attacks. In addition, lower-layer protocols need to describe what attributes should be included in channel binding requests. EAP methods need to be updated in order to describe how the channel binding request and response are carried. In addition, deployments may need to decide what information is populated in the local database. The following sections describe types of attacks that can be prevented by this framework with appropriate lower-layer attributes carried in channel bindings, EAP methods with channel binding support and appropriate local database information at the EAP server.

A.1. Enterprise Subnetwork Masquerading

As outlined in Section 3, an enterprise network may have multiple VLANs providing different levels of security. In an attack, a malicious NAS connecting to a guest network with lesser security protection could broadcast the SSID of a subnetwork with higher protection. This could lead peers to believe that they are accessing the network over secure connections, and, e.g., transmit confidential information that they normally would not send over a weakly protected connection. This attack works under the conditions that peers use the same set of credentials to authenticate to the different kinds of VLANs and that the VLANs support at least one common EAP method. If these conditions are not met, the EAP server would not authorize the peers to connect to the guest network, because the peers used credentials and/or an EAP method that is associated with the corporate network.

A.2. Forced Roaming

Mobile phone providers boosting their cell tower's transmission power to get more users to use their networks have occurred in the past. The increased transmission range combined with a NAS sending a false network identity lures users to connect to the network without being aware of that they are roaming.

Channel bindings would detect the bogus network identifier because the network identifier sent to the authentication server in `il` will neither match information `i2` nor the stored data. The verification fails because the info in `il` claims to come from the peer's home network while the home authentication server knows that the connection is through a visited network outside the home domain. In the same context, channel bindings can be utilized to provide a "home zone" feature that notifies users every time they are about to connect to a NAS outside their home domain.

A.3. Downgrading attacks

A malicious authenticator could modify the set of offered EAP methods in its Beacon to force the peer to choose from only the weakest EAP method(s) accepted by the authentication server. For instance, instead of having a choice between EAP-MD5-CHAP, EAP-FAST and some other methods, the authenticator reduces the choice for the peer to the weaker EAP-MD5-CHAP method. Assuming that weak EAP methods are supported by the authentication server, such a downgrading attack can enable the authenticator to attack the integrity and confidentiality of the remaining EAP execution and/or break the authentication and key exchange. The presented channel bindings prevent such downgrading attacks, because peers submit the offered EAP method

selection that they have received in the beacon as part of il to the authentication server. As a result, the authentication server recognizes the modification when comparing the information to the respective information in its policy database. This presumes that all acceptable EAP methods support channel binding and that an attacker cannot break the EAP method in real-time.

A.4. Bogus Beacons in IEEE 802.11r

In IEEE 802.11r, the SSID is bound to the TSK calculations, so that the TSK needs to be consistent with the SSID advertised in an authenticator's Beacon. While this prevents outsiders from spoofing a Beacon it does not stop a "lying NAS" from sending a bogus Beacon and calculating the TSK accordingly.

By implementing channel bindings, as described in this draft, in IEEE 802.11r, the verification by the authentication server would detect the inconsistencies between the information the authenticator has sent to the peer and the information the server received from the authenticator and stores in the policy database.

A.5. Forcing false authorization in IEEE 802.11i

In IEEE 802.11i a malicious NAS can modify the beacon to make the peer believe it is connected to a network different from the one the peer is actually connected to.

In addition, a malicious NAS can force an authentication server into authorizing access by sending an incorrect Called-Station-ID that belongs to an authorized NAS in the network. This could cause the authentication server to believe it had granted access to a different network or even provider than the one the peer got access to.

Both attacks can be prevented by implementing channel bindings, because the server can compare the information that was sent to the peer, with information it received from the authenticator during the AAA communication as well as the information stored in the policy database.

Appendix B. Change History

RFC editor, remove this section prior to publication.

B.1. Changes Since 09

Based on WG discussion, all assigned numbers start at 1, including NSIDs and codes.

Based on WG discussion we include the value of attributes in the RADIUS namespace in channel binding responses.

B.2. Changes since Version 06

The purpose of this revision is to provide a specific candidate protocol for channel binding data and channel binding responses.

B.3. Changes since version 04

- o Clarify examples in introduction.
- o In problem statement note that one EAP server may deal with both enterprise and provider networks.
- o Update discussion of the architecture. Talk about channel bindings as a mechanism to introduce levels of trust.
- o Indicate that this document is focusing on EAP channel bindings within methods while trying to do a better job of describing the SAP approach in more detail.
- o Claim that we're using the encoding from draft-clancy-emu-aaapay. The WG almost certainly doesn't have consensus on this, but in the interest of actually describing what the protocol might be like, it is a good straw-man proposal.
- o Update protocol description.

Authors' Addresses

Sam Hartman (editor)
Painless Security
356 Abbott ST
North Andover, MA 01845
USA

Email: hartmans-ietf@mit.edu

T. Charles Clancy
Department of Electrical Engineering and Computer Science
Virginia Tech
Arlington, VA 22203
USA

Email: tcc@vt.edu

Katrin Hoeper
Motorola, Inc.
1301 E. Algonquin Road
Schaumburg, IL 60196
USA

Email: khoeper@motorolasolutions.com

EMU Working Group
Internet-Draft
Intended status: Informational
Expires: June 19, 2011

K. Hoeper
Motorola, Inc.
S. Hanna
Juniper Networks
H. Zhou
J. Salowey, Ed.
Cisco Systems, Inc.
December 16, 2010

Requirements for a Tunnel Based EAP Method
draft-ietf-emu-eaptunnel-req-09.txt

Abstract

This memo defines the requirements for a tunnel-based Extensible Authentication Protocol (EAP) Method. This tunnel method will use Transport Layer Security (TLS) to establish a secure tunnel. The tunnel will provide support for password authentication, EAP authentication and the transport of additional data for other purposes.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

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."

This Internet-Draft will expire on June 19, 2011.

Copyright Notice

Copyright (c) 2010 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Introduction	5
2. Conventions Used In This Document	5
3. Use Cases	6
3.1. Password Authentication	6
3.2. Protection of Weak EAP Methods	6
3.3. Chained EAP Methods	7
3.4. Identity Protection	7
3.5. Anonymous Service Access	7
3.6. Network Endpoint Assessment	8
3.7. Client Authentication During Tunnel Establishment	8
3.8. Extensibility	8
3.9. Certificate-less Authentication and Generic EAP Method Extension	9
4. Requirements	9
4.1. General Requirements	9
4.1.1. RFC Compliance	9
4.2. Tunnel Requirements	10
4.2.1. TLS Requirements	10
4.2.1.1. Cipher Suites	10
4.2.1.1.1. Cipher Suite Negotiation	10
4.2.1.1.2. Tunnel Data Protection Algorithms	11
4.2.1.1.3. Tunnel Authentication and Key Establishment	11
4.2.1.2. Tunnel Replay Protection	12
4.2.1.3. TLS Extensions	12
4.2.1.4. Peer Identity Privacy	12
4.2.1.5. Session Resumption	12
4.2.2. Fragmentation	12
4.2.3. Protection of Data External to Tunnel	13
4.3. Tunnel Payload Requirements	13
4.3.1. Extensible Attribute Types	13
4.3.2. Request/Challenge Response Operation	13
4.3.3. Indicating Criticality of Attributes	13
4.3.4. Vendor Specific Support	13
4.3.5. Result Indication	14
4.3.6. Internationalization of Display Strings	14
4.4. EAP Channel Binding Requirements	14
4.5. Requirements Associated with Carrying Username and Passwords	14
4.5.1. Security	15
4.5.1.1. Confidentiality and Integrity	15
4.5.1.2. Authentication of Server	15
4.5.1.3. Server Certificate Revocation Checking	15
4.5.2. Internationalization	15

4.5.3. Meta-data	16
4.5.4. Password Change	16
4.6. Requirements Associated with Carrying EAP Methods	16
4.6.1. Method Negotiation	16
4.6.2. Chained Methods	16
4.6.3. Cryptographic Binding with the TLS Tunnel	16
4.6.4. Peer Initiated	18
4.6.5. Method Meta-data	18
5. IANA Considerations	18
6. Security Considerations	18
6.1. Cipher Suite Selection	18
6.2. Tunneled Authentication	19
6.3. Data External to Tunnel	20
6.4. Separation of TLS Tunnel and Inner Authentication Termination	20
7. References	20
7.1. Normative References	20
7.2. Informative References	21
Appendix A. Changes from -01	23
Appendix B. Changes from -02	23
Appendix C. changes from -03	23
Authors' Addresses	24

1. Introduction

An Extensible Authentication Protocol (EAP) tunnel method is an EAP method that establishes a secure tunnel and executes other EAP methods under the protection of that secure tunnel. An EAP tunnel method can be used in any lower layer protocol that supports EAP authentication. There are several existing EAP tunnel methods that use Transport Layer Security (TLS) to establish the secure tunnel. EAP methods supporting this include Protected EAP (PEAP) [PEAP], Tunneled Transport Layer Security EAP (TTLS) [RFC5281] and EAP Flexible Authentication via Secure Tunneling (EAP-FAST) [RFC4851]. In general this has worked well so there is consensus to continue to use TLS as the basis for a tunnel method. There have been various reasons for employing a protected tunnel for EAP processes. They include protecting weak authentication exchanges, such as username and password. In addition a protected tunnel can provide means to provide peer identity protection and EAP method chaining. Finally, systems have found it useful to transport additional types of data within the protected tunnel.

This document describes the requirements for a EAP tunnel method as well as for a password protocol supporting legacy password verification within the tunnel method.

2. Conventions Used In This Document

Use of each capitalized word within a sentence or phrase carries the following meaning during the EAP Method Update (EMU) WG's method selection process:

MUST - indicates an absolute requirement

MUST NOT - indicates something absolutely prohibited

SHOULD - indicates a strong recommendation of a desired result

SHOULD NOT - indicates a strong recommendation against a result

MAY - indicates a willingness to allow an optional outcome

Lower case uses of "MUST", "MUST NOT", "SHOULD", "SHOULD NOT" and "MAY" carry their normal meaning and are not subject to these definitions.

3. Use Cases

To motivate and explain the requirements in this document, a representative set of use cases for the EAP tunnel method are supplied here. It is mandatory for a candidate tunnel method to support all of the use cases that are marked below as "MUST".

3.1. Password Authentication

Many legacy systems only support user authentication with passwords. Some of these systems require transport of the actual username and password to the authentication server. This is true for systems where the authentication server does not have access to the cleartext password or a consistent transform of the cleartext password. Example of such systems are some one time password (OTP) systems and other systems where the username and password are submitted to an external party for validation. The tunnel method MUST support transporting cleartext username and password to the EAP server. It MUST NOT reveal information about the username and password to parties in the communication path between the peer and the EAP Server. The advantage any attacker gains against the tunnel method when employing a username and password for authentication MUST be through interaction and not computation. The tunnel MUST support protection from man-in-the-middle attacks. The combination of the tunnel authentication and password authentication MUST enable mutual authentication.

Since EAP authentication occurs before network access is granted the tunnel method SHOULD enable an inner exchange to provide support for minimal password management tasks including password change, "new PIN mode", and "next token mode" required by some systems.

3.2. Protection of Weak EAP Methods

Some existing EAP methods have vulnerabilities that could be eliminated or reduced by running them inside a protected tunnel. For example, a EAP-MD5 does not provide mutual authentication or protection from dictionary attacks. Without extra protection, EAP tunnel methods are vulnerable to a special type of tunnel man-in-the-middle attack [TUNNEL-MITM]. This attack is referred to as "tunnel MitM attack" in the remainder of this document. The additional protection needed to thwart tunnel MitM attacks depends on the inner method executed within the tunnel. When weak methods are used, these attacks can be mitigated via security policies that require the method to be used only within a tunnel. On the other hand, a technical solution (so-called cryptographic bindings) can be used whenever the inner method derives key material and is not susceptible to attacks outside a tunnel. Only the latter mitigation technique

can be made an actual requirement for EAP tunnel methods (see Section 4.6.3), while security policies are outside the scope of this requirement draft. Please refer to the NIST Recommendation for EAP Methods Used in Wireless Network Access Authentication [NIST SP 800-120] and [LCN 2010] for a discussion on security policies and complete solutions for thwarting tunnel MitM attacks.

The tunnel method MUST support protection of weak EAP methods. Cryptographic protection from tunnel MitM attacks MUST be provided for all key generating methods. In combination with an appropriate security policy this will thwart MitM attacks against inner methods.

3.3. Chained EAP Methods

Several circumstances are best addressed by using chained EAP methods. For example, it may be desirable to authenticate the user and also authenticate the device being used. However, chained EAP methods from different conversations can be re-directed into the same conversation by an attacker giving the authenticator the impression that both conversations terminate at the same end-point. Cryptographic binding can be used to bind the results of chained key generating methods together or to an encompassing tunnel.

The tunnel method MUST support chained EAP methods while including protection against attacks on method chaining.

3.4. Identity Protection

When performing an EAP authentication, the peer may want to protect its identity and only disclose it to a trusted EAP server. This helps to maintain peer privacy.

The tunnel method MUST support identity protection, therefore the identity of the peer used for authentication purposes MUST NOT be obtainable by any entity other than the EAP server terminating the tunnel method. Peer identity protection provided by the tunnel method applies to tunnel method and inner method specific identities. Note that the peer may need to expose the realm portion of the EAP outer identity in the Network Access Identifier (NAI) [RFC4282] in a roaming scenario in order to reach the appropriate authentication server.

3.5. Anonymous Service Access

When network service is provided, it is sometimes desirable for a user to gain network access in order to access limited services for emergency communication or troubleshooting information. To avoid eavesdropping, it's best to negotiate link layer security as with any

other authentication.

Therefore, the tunnel method SHOULD allow anonymous peers or server-only authentication, while still deriving keys that can be used for link layer security. The tunnel method MAY also allow for the bypass of server authentication processing on the client.

Forgoing user or server authentication increases the chance of man-in-the-middle and other types of attacks that can compromise the derived keys used for link layer security. Therefore, passwords and other sensitive information MUST NOT be disclosed to an unauthenticated server, or to a server that is not authorized to authenticate the user.

3.6. Network Endpoint Assessment

The Network Endpoint Assessment (NEA) protocols and reference model described in [RFC5209] provide a standard way to check the health ("posture") of a device at or after the time it connects to a network. If the device does not comply with the network's requirements, it can be denied access to the network or granted limited access to remediate itself. EAP is a convenient place for conducting an NEA exchange.

The tunnel method SHOULD support carrying NEA protocols such as PB-TNC [RFC5793]. Depending on the specifics of the tunnel method, these protocols may be required to be carried in an EAP method.

3.7. Client Authentication During Tunnel Establishment

In some cases the peer will have credentials that allow it to authenticate during tunnel establishment. These credentials may only partially authenticate the identity of the peer and additional authentication may be required inside the tunnel. For example, a communication device may be authenticated during tunnel establishment, in addition user authentication may be required to satisfy authentication policy. The tunnel method MUST be capable of providing client side authentication during tunnel establishment.

3.8. Extensibility

The tunnel method MUST provide extensibility so that additional data related to authentication, authorization and network access can be carried inside the tunnel in the future. This removes the need to develop new tunneling methods for specific purposes.

An application for extensibility is credential provisioning. When a peer has authenticated with EAP, this is a convenient time to

distribute credentials to that peer that may be used for later authentication exchanges. For example, the authentication server can provide a private key or shared key to the peer that can be used by the peer to perform rapid re-authentication or roaming. In addition there have been proposals to perform enrollment within EAP, such as [I-D.mahy-eap-enrollment]. Another use for extensibility is support for alternate authentication frameworks within the tunnel.

3.9. Certificate-less Authentication and Generic EAP Method Extension

In some cases the peer will not have a way to verify a server certificate and in some cases a server might not have a certificate to verify. Therefore, it is desirable to support certificate-less authentication. An application for this is credential provisioning where the peer and server authenticate each other with a shared password and credentials for subsequent authentication (e.g. a key pair and certificate or a shared key) can be passed inside the tunnel. Another application is to extend existing EAP methods with new features such as EAP channel bindings.

Great care must be taken when using tunnels with no server authentication for the protection of an inner method. For example, the client may lack the appropriate trust roots to fully authenticate the server, but may still establish the tunnel to execute an inner EAP method to perform mutual authentication and key derivation. In these cases, the inner EAP method **MUST** provide resistance to dictionary attack and a cryptographic binding between the inner method and the tunnel method **MUST** be established. Furthermore, the cipher suite used to establish the tunnel **MUST** derive the master key using contribution from both client and server, as in ephemeral Diffie-Hellman cipher suites.

The tunnel method **MAY** allow for certificate-less authentication.

4. Requirements

4.1. General Requirements

4.1.1. RFC Compliance

The tunnel method **MUST** include a Security Claims section with all security claims specified in Section 7.2 in RFC 3748 [RFC3748]. In addition, it **MUST** meet the requirement in Sections 2.1 and 7.4 of RFC 3748 that tunnel methods **MUST** support protection against man-in-the-middle attacks. Furthermore, the tunnel method **MUST** support identity protection as specified in Section 7.3 in RFC 3748.

The tunnel method MUST be unconditionally compliant with RFC 4017 [RFC4017] (using the definition of "unconditionally compliant" contained in section 1.1 of RFC 4017). This means that the method MUST satisfy all the MUST, MUST NOT, SHOULD, and SHOULD NOT requirements in RFC 4017.

The tunnel method MUST meet all the MUST and SHOULD requirements relevant to EAP methods contained in the EAP Key Management Framework [RFC5247] or any successor. This includes the generation of the MSK, EMSK, Peer-Id, Server-Id and Session-Id. These requirements will enable the tunnel method to properly fit into the EAP key management framework, maintaining all of the security properties and guarantees of that framework.

The tunnel method MUST NOT be tied to any single cryptographic algorithm. Instead, it MUST support run-time negotiation to select among an extensible set of cryptographic algorithms, such as algorithms used with certificates presented during tunnel establishment. This "cryptographic algorithm agility" provides several advantages. Most important, when a weakness in an algorithm is discovered or increased processing power overtakes an algorithm, users can easily transition to a new algorithm. Also, users can choose the algorithm that best meets their needs.

The tunnel method MUST meet the SHOULD and MUST requirements pertinent to EAP method contained in Section 3 of RFC 4962 [RFC4962]. This includes: cryptographic algorithm independence; strong, fresh session keys; replay detection; keying material confidentiality and integrity; and confirmation of cipher suite selection.

4.2. Tunnel Requirements

The following section discusses requirements for TLS Tunnel Establishment.

4.2.1. TLS Requirements

The tunnel based method MUST support TLS version 1.2 [RFC5246] and may support earlier versions greater than SSL 2.0 to enable the possibility of backwards compatibility.

4.2.1.1. Cipher Suites

4.2.1.1.1. Cipher Suite Negotiation

Cipher suite negotiations always suffer from downgrading attacks when they are not secured by any kind of integrity protection. A common practice is a post integrity check in which, as soon as available,

the established keys (here the tunnel key) are used to derive integrity keys. These integrity keys are then used by peer and authentication server to verify whether the cipher suite negotiation has been maliciously altered by another party.

Integrity checks prevent downgrading attacks only if the derived integrity keys and the employed integrity algorithms cannot be broken in real-time. See Section 6.1 or [KHL07] for more information on this. Hence, the tunnel method MUST provide integrity protected cipher suite negotiation with secure integrity algorithms and integrity keys.

TLS provides protected cipher suite negotiation as long as all the cipher suites supported provide authentication, key establishment and data integrity protection as discussed in Section 6.1.

4.2.1.1.2. Tunnel Data Protection Algorithms

In order to prevent attacks on the cryptographic algorithms employed by inner authentication methods, a tunnel protocol's protection needs to provide a basic level of algorithm strength. The tunnel method MUST provide at least one mandatory to implement cipher suite that provides the equivalent security of 128-bit AES for encryption and message authentication. See Part 1 of the NIST Recommendation for Key Management [NIST SP 800-57] for a discussion of the relative strengths of common algorithms.

4.2.1.1.3. Tunnel Authentication and Key Establishment

A tunnel method MUST provide unidirectional authentication from authentication server to EAP peer and mutual authentication between authentication server and EAP peer. The tunnel method MUST provide at least one mandatory to implement cipher suite that provides certificate-based authentication of the server and provides optional certificate-based authentication of the client. Other types of authentication MAY be supported.

At least one mandatory to implement cipher suite MUST be approved by NIST Draft Recommendation for Key Management, Part 3 [NIST SP 800-57p3], i.e., the ciphersuite MUST be listed in Table 4-1, 4-2 or 4-3 in that document.

The mandatory to implement cipher suites MUST only include cipher suites that use strong cryptographic algorithms. They MUST NOT include cipher suites providing mutually anonymous authentication or static Diffie-Hellman cipher suites.

Other ciphersuites MAY be selected following the security

requirements for tunnel protocols in NIST DRAFT Recommendation for EAP Methods Used in Wireless Network Access Authentication [NIST SP 800-120].

4.2.1.2. Tunnel Replay Protection

In order to prevent replay attacks on a tunnel protocol, the message authentication MUST be generated using a time-variant input such as timestamps, sequence numbers, nonces, or a combination of these so that any re-use of the authentication data can be detected as invalid. TLS provides sufficient replay protection to meet this requirements as long as weak cipher suites discussed in Section 6.1 are avoided.

4.2.1.3. TLS Extensions

In order to meet the requirements in this document TLS extensions MAY be used. For example, TLS extensions may be useful in providing certificate revocation information via the TLS Online Certificate Status Protocol (OCSP) extension [I-D.ietf-tls-rfc4366-bis] (thus meeting the requirement in Section 4.5.1.3).

4.2.1.4. Peer Identity Privacy

A tunnel protocol MUST support peer privacy. This requires that the username and other attributes associated with the peer are not transmitted in the clear or to an unauthenticated, unauthorized party. Peer identity protection provided by the tunnel method applies to establishment of the tunnel and protection of inner method specific identities. If applicable, the peer certificate is sent confidentially (i.e. encrypted).

4.2.1.5. Session Resumption

The tunnel method MUST support TLS session resumption as defined in [RFC5246]. The tunnel method MAY support other methods of session resumption such as those defined in [RFC5077].

4.2.2. Fragmentation

Tunnel establishment sometimes requires the exchange of information that exceeds what can be carried in a single EAP message. In addition information carried within the tunnel may also exceed this limit. Therefore a tunnel method MUST support fragmentation and reassembly.

4.2.3. Protection of Data External to Tunnel

A man-in-the-middle attacker can modify clear text values such as protocol version and type code information communicated outside the TLS tunnel. The tunnel method MUST provide implicit or explicit protection of the protocol version and type code. If modification of other information external to the tunnel can cause exploitable vulnerabilities, the tunnel method MUST provide protection against modification of this additional data.

4.3. Tunnel Payload Requirements

This section describes the payload requirements inside the tunnel. These requirements frequently express features that a candidate protocol must be capable of offering so that a deployer can decide whether to make use of that feature. This section does not state requirements about what features of each protocol must be used during a deployment.

4.3.1. Extensible Attribute Types

The payload MUST be extensible. Some standard payload attribute types will be defined to meet known requirements listed below, such as password authentication, inner EAP method, vendor specific attributes, and result indication. Additional payload attributes MAY be defined in the future to support additional features and data types.

4.3.2. Request/Challenge Response Operation

The payload MUST support request and response type of half-duplex operation typical of EAP. Multiple attributes may be sent in a single payload. The payload MAY support carrying on multiple authentications in a single payload packet.

4.3.3. Indicating Criticality of Attributes

It is expected that new attributes will be defined to be carried within the tunnel method. In some cases it is necessary for the sender to know if the receiver did not understand the attribute. To support this, there MUST be a way for the sender to mark attributes such that the receiver will indicate if an attribute is not understood.

4.3.4. Vendor Specific Support

The payload MUST support communication of an extensible set of vendor-specific attributes. These attributes will be segmented into

uniquely identified vendor specific name spaces. They can be used for experiments or vendor specific features.

4.3.5. Result Indication

The payload MUST support result indication and its acknowledgement, so both the EAP peer and server will end up with a synchronized state. The result indication is needed after each chained inner authentication method and at the end of the authentication, so separate result indication for intermediate and final result MUST be supported.

4.3.6. Internationalization of Display Strings

The payload MAY provide a standard attribute format that supports international strings. This attribute format MUST support encoding strings in UTF-8 [RFC3629] format. Any strings sent by the server intended for display to the user MUST be sent in UTF-8 format and SHOULD be able to be marked with language information and adapted to the user's language preference as indicated by RFC 5646 [RFC5646]. Note that in some cases, such as when transmitting error codes, it is acceptable to exchange numeric codes that can be translated by the client to support the particular local language. These numeric codes are not subject internationalization during transmission.

4.4. EAP Channel Binding Requirements

The tunnel method MUST be capable of meeting EAP channel binding requirements described in [I-D.ietf-emu-chbind]. As discussed in [RFC5056], EAP Channel bindings differ from channel bindings discussed in other contexts. Cryptographic binding between the TLS tunnel and the inner method discussed in Section 4.6.3 relates directly to the non-EAP channel binding concepts discussed in RFC 5056.

4.5. Requirements Associated with Carrying Username and Passwords

This section describes the requirements associated with tunneled password authentication. The password authentication mentioned here refers to user or machine authentication using a legacy password database or verifier, such as LDAP [RFC4511], OTP, etc. These typically require the password in its original text form in order to authenticate the peer, hence they require the peer to send the clear text user name and password to the EAP server.

4.5.1. Security

Many internal EAP methods have the peer send its password in the clear to the EAP server. Other methods (e.g. challenge-response methods) are vulnerable to attacks if an eavesdropper can intercept the traffic. For any such methods, the security measures in the following sections MUST be met.

4.5.1.1. Confidentiality and Integrity

The clear text password exchange MUST be integrity and confidentiality protected. As long as the password exchange occurs inside an authenticated and encrypted tunnel, this requirement is met.

4.5.1.2. Authentication of Server

The EAP server MUST be authenticated before the peer sends the clear text password to the server.

4.5.1.3. Server Certificate Revocation Checking

When certificate authentication is used during tunnel establishment the EAP peer may need to present its password to the server before it has network access to check the revocation status of the server's credentials. Therefore, the tunnel method MUST support mechanisms to check the revocation status of a credential. The tunnel method SHOULD make use of Online Certificate Status Protocol (OCSP) [RFC2560] or Server-based Certificate Validation Protocol (SCVP) [RFC5055] to obtain the revocation status of the EAP server certificate.

4.5.2. Internationalization

The password authentication exchange MUST support user names and passwords in international languages. It MUST support encoding of user name and password strings in UTF-8 [RFC3629] format. The method MUST specify how username and password normalizations and/or comparisons is performed in reference to SASLPrep [RFC4013], Net-UTF-8 [RFC5198] or their replacement.

Any strings sent by the server intended for display to the user MUST be sent in UTF-8 format and SHOULD be able to be marked with language information and adapted to the user's language preference as indicated by RFC 5646 [RFC5646]. Note that in some cases, such as when transmitting error codes, it is acceptable to exchange numeric codes that can be translated by the client to support the particular local language. These numeric codes are not subject to

internationalization during transmission.

4.5.3. Meta-data

The password authentication exchange SHOULD support additional associated meta-data which can be used to indicate whether the authentication is for a user or a machine. This allows the EAP server and peer to request and negotiate authentication specifically for a user or machine. This is useful in the case of multiple inner authentications where the user and machine both need to be authenticated.

4.5.4. Password Change

The password authentication exchange MUST support password change. The exchange SHOULD be extensible to support other "housekeeping" functions, such as the management of PINs or other data, required by some systems.

4.6. Requirements Associated with Carrying EAP Methods

The tunnel method MUST be able to carry inner EAP methods without modifying them. EAP methods MUST NOT be redefined inside the tunnel.

4.6.1. Method Negotiation

The tunnel method MUST support the protected negotiation of the inner EAP method. It MUST NOT allow the inner EAP method negotiation to be manipulated by intermediaries.

4.6.2. Chained Methods

The tunnel method SHOULD support the chaining of multiple EAP methods. The tunnel method MUST allow for the communication of intermediate result and verification of compound binding between executed inner methods when chained methods are employed.

4.6.3. Cryptographic Binding with the TLS Tunnel

The tunnel method MUST provide a mechanism to bind the tunnel protocol and the inner EAP method. This property is referred to as cryptographic binding. Such bindings are an important tool for mitigating the tunnel MitM attacks on tunnel methods [TUNNEL-MITM]. Cryptographic bindings enable the complete prevention of tunnel MitM attacks without the need of additional security policies as long as the inner method derives keys and is not vulnerable to attacks outside a protected tunnel [LCN 2010]. Even though weak or non-key deriving inner methods may be permitted, and thus security policies

preventing tunnel MitM attacks are still necessary, the tunnel method MUST provide cryptographic bindings, because only this allows migrating to more secure, policy-independent implementations.

Cryptographic bindings are typically achieved by securely mixing the established keying material (say tunnel key TK) from the tunnel protocol with the established keying material (say method key MK) from the inner authentication method(s) in order to derive fresh keying material. If chained EAP methods are executed in the tunnel, all derived inner keys are combined with the tunnel key to create a new compound tunnel key (CTK). In particular, CTK is used to derive the EAP MSK, EMSK and other transient keys (TEK), such as transient encryption keys and integrity protection keys. The key hierarchy for tunnel methods executions that derive compound keys for the purpose of cryptographic binding is depicted in Figure 1.

In the case of the sequential executions of n inner methods, a chained compound key CTK_i MUST be computed upon the completion of each inner method i such that it contains the compound key of all previous inner methods, i.e. $CTK_i = f(CTK_{i-1}, MK_i)$ with $0 < i \leq n$ and $CTK_0 = TK$, where $f()$ is a key derivation function, such as one that complies with NIST Recommendation for Key Derivation Using Pseudorandom Functions [NIST SP 800-108]. CTK_n SHOULD serve as the key to derive further keys. Figure 1 depicts the key hierarchy in the case of a single inner method. Transient keys derived from the compound key CTK are used in a cryptographic protocol to verify the integrity of the tunnel and the inner authentication method.

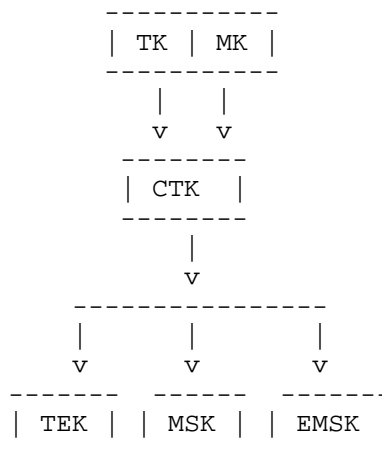


Figure 1: Compound Keys

Furthermore, all compound keys CTK_i and all keys derived from it

SHOULD follow the recommendations for key derivations and key hierarchies as specified in [NIST SP 800-108]. In particular, all derived keys MUST have a lifetime assigned that does not exceed the lifetime of any key higher in the key hierarchy. The derivation MUST prevent a compromise in one part of the system from leading to compromises in other parts of the system that relay on keys at the same or higher level in the hierarchy.

4.6.4. Peer Initiated

The tunnel method SHOULD allow for the peer to initiate an inner EAP authentication in order to meet its policy requirements for authenticating the server.

4.6.5. Method Meta-data

The tunnel method SHOULD allow for the communication of additional data associated with an EAP method. This can be used to indicate whether the authentication is for a user or a machine. This allows the EAP server and peer to request and negotiate authentication specifically for a user or machine. This is useful in the case of multiple inner EAP authentications where the user and machine both need to be authenticated.

5. IANA Considerations

This document has no IANA considerations.

6. Security Considerations

A tunnel method is often deployed to provide mutual authentication between EAP Peer and EAP Server and to generate key material for use in protecting lower layer protocols. In addition the tunnel is used to protect the communication of additional data, including peer identity between the EAP Peer and EAP Server from disclosure to or modification by an attacker. These sections cover considerations that affect the ability for a method to achieve these goals.

6.1. Cipher Suite Selection

TLS supports a wide variety of cipher suites providing a variety of security properties. The selection of cipher suites is critical to the security of the tunnel method. Selection of a cipher suite with weak or no authentication, such as an anonymous Diffie-Hellman based cipher suite will greatly increase the risk of system compromise. Since a tunnel method uses the TLS tunnel to transport data, the

selection of a ciphersuite with weak data encryption and integrity algorithms will also increase the vulnerability of the method to attacks.

A tunnel protocol is prone to downgrading attacks if the tunnel protocol supports any key establishment algorithm that can be broken on-line. In a successful downgrading attack, an adversary breaks the selected "weak" key establishment algorithm and optionally the "weak" authentication algorithm without being detected. Here, "weak" refers to a key establishment algorithm that can be broken in real-time, and an authentication scheme that can be broken off-line, respectively. See [KHLCO7] for more details. The requirements in this document disapprove the use of key establishment algorithms that can be broken on-line.

Mutually anonymous tunnel protocols are prone to man-in-the-middle attacks described in [KHLCO7]. During such an attack, an adversary establishes a tunnel with each the peer and the authentication server, while peer and server believe that they established a tunnel with each other. Once both tunnels have been established, the adversary can eavesdrop on all communications within the tunnels, i.e. the execution of the inner authentication method(s). Consequently, the adversary can eavesdrop on the identifiers that are exchanged as part of the EAP method and thus, the privacy of peer and/or authentication server is compromised along with any other data transmitted within the tunnels. This document requires server authentication to avoid the risks associated with anonymous cipher suites.

6.2. Tunneled Authentication

In many cases a tunnel method provides mutual authentication by authenticating the server during tunnel establishment and authenticating the peer within the tunnel using an EAP method. As described in [TUNNEL-MITM], this mode of operation can allow tunnel man-in-the-middle attackers to authenticate to the server as the peer by tunneling the inner EAP protocol messages to and from a peer executing the method outside a tunnel or with an untrustworthy server. Cryptographic binding between the established keying material from the inner authentication method(s) and the tunnel protocol verifies that the endpoints of the tunnel and the inner authentication method(s) are the same. This can thwart the attack if the inner method derived keys of sufficient strength that they cannot be broken in real-time.

In cases where the inner authentication method does not generate any or only weak key material, security policies MUST be enforced such that the peer cannot execute the inner method with the same

credentials outside a protective tunnel or with an untrustworthy server.

6.3. Data External to Tunnel

The tunnel method will use data that is outside the TLS tunnel such as the EAP type code or version numbers. If an attacker can compromise the protocol by modifying these values the tunnel method MUST protect this data from modification. In some cases external data may not need additional protection because it is implicitly verified during the protocol operation.

6.4. Separation of TLS Tunnel and Inner Authentication Termination

Terminating the inner method at a different location than the outer tunnel needs careful consideration. The inner method data may be vulnerable to modification and eavesdropping between the server that terminates the tunnel and the server that terminates the inner method. For example if a clear text password is used then it may be sent to the inner method server in a RADIUS password attribute which uses weak encryption that may not be suitable protection for many environments.

In some cases terminating the tunnel at a different location may make it difficult for a peer to authenticate the server and trust it for further communication. For example, if the TLS tunnel is terminated by a different organization the peer needs to be able to authenticate and authorize the tunnel server to handle secret credentials that it shares with the home server that terminates the inner method. This may not meet the security policy of many environments.

7. References

7.1. Normative References

- [I-D.ietf-emu-chbind]
Hartman, S., Clancy, C., and K. Hoeper, "Channel Binding Support for EAP Methods", draft-ietf-emu-chbind-06 (work in progress), October 2010.
- [RFC2560] Myers, M., Ankney, R., Malpani, A., Galperin, S., and C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP", RFC 2560, June 1999.
- [RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

- [RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowitz, "Extensible Authentication Protocol (EAP)", RFC 3748, June 2004.
- [RFC4017] Stanley, D., Walker, J., and B. Aboba, "Extensible Authentication Protocol (EAP) Method Requirements for Wireless LANs", RFC 4017, March 2005.
- [RFC4962] Housley, R. and B. Aboba, "Guidance for Authentication, Authorization, and Accounting (AAA) Key Management", BCP 132, RFC 4962, July 2007.
- [RFC5055] Freeman, T., Housley, R., Malpani, A., Cooper, D., and W. Polk, "Server-Based Certificate Validation Protocol (SCVP)", RFC 5055, December 2007.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.
- [RFC5247] Aboba, B., Simon, D., and P. Eronen, "Extensible Authentication Protocol (EAP) Key Management Framework", RFC 5247, August 2008.

7.2. Informative References

- [I-D.ietf-tls-rfc4366-bis]
3rd, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", draft-ietf-tls-rfc4366-bis-12 (work in progress), September 2010.
- [I-D.mahy-eap-enrollment]
Mahy, R., "An Extensible Authentication Protocol (EAP) Enrollment Method", draft-mahy-eap-enrollment-01 (work in progress), March 2006.
- [KHLCO7] Hoeper, K. and L. Chen, "Where EAP Security Claims Fail", ICST QShine , August 2007.
- [LCN 2010]
Hoeper, K. and L. Chen, "An Inconvenient Truth about Tunneled Authentications", Proceedings of 35th Annual IEEE Conference on Local Computer Networks (LCN 2010) , September 2009.
- [NIST SP 800-108]
Chen, L., "Recommendation for Key Derivation Using Pseudorandom Functions", Draft NIST Special Publication 800-108, April 2008.

- [NIST SP 800-120]
Hoeper, K. and L. Chen, "Recommendation for EAP Methods Used in Wireless Network Access Authentication", NIST Special Publication 800-120, September 2009.
- [NIST SP 800-57]
Barker, E., Barker, W., Burr, W., Polk, W., and M. Smid, "Recommendation for Key Management - Part 1: General (Revised)", NIST Special Publication 800-57, part 1, March 2007.
- [NIST SP 800-57p3]
Barker, E., Burr, W., Jones, A., Polk, W., , S., and M. Smid, "Recommendation for Key Management, Part 3 Application-Specific Key Management Guidance", Draft NIST Special Publication 800-57, part 3, October 2008.
- [PEAP]
Microsoft Corporation, "[MS-PEAP]: Protected Extensible Authentication Protocol (PEAP) Specification", August 2009.
- [RFC4013]
Zeilenga, K., "SASLprep: Stringprep Profile for User Names and Passwords", RFC 4013, February 2005.
- [RFC4282]
Aboba, B., Beadles, M., Arkko, J., and P. Eronen, "The Network Access Identifier", RFC 4282, December 2005.
- [RFC4511]
Sermersheim, J., "Lightweight Directory Access Protocol (LDAP): The Protocol", RFC 4511, June 2006.
- [RFC4851]
Cam-Winget, N., McGrew, D., Salowey, J., and H. Zhou, "The Flexible Authentication via Secure Tunneling Extensible Authentication Protocol Method (EAP-FAST)", RFC 4851, May 2007.
- [RFC5056]
Williams, N., "On the Use of Channel Bindings to Secure Channels", RFC 5056, November 2007.
- [RFC5077]
Salowey, J., Zhou, H., Eronen, P., and H. Tschofenig, "Transport Layer Security (TLS) Session Resumption without Server-Side State", RFC 5077, January 2008.
- [RFC5198]
Klensin, J. and M. Padlipsky, "Unicode Format for Network Interchange", RFC 5198, March 2008.
- [RFC5209]
Sangster, P., Khosravi, H., Mani, M., Narayan, K., and J. Tardo, "Network Endpoint Assessment (NEA): Overview and Requirements", RFC 5209, June 2008.

- [RFC5281] Funk, P. and S. Blake-Wilson, "Extensible Authentication Protocol Tunneled Transport Layer Security Authenticated Protocol Version 0 (EAP-TTLSv0)", RFC 5281, August 2008.
- [RFC5646] Phillips, A. and M. Davis, "Tags for Identifying Languages", BCP 47, RFC 5646, September 2009.
- [RFC5793] Sahita, R., Hanna, S., Hurst, R., and K. Narayan, "PB-TNC: A Posture Broker (PB) Protocol Compatible with Trusted Network Connect (TNC)", RFC 5793, March 2010.
- [TUNNEL-MITM] Asokan, N., Niemi, V., and K. Nyberg, "Man-in-the-Middle in Tunnelled Authentication Protocols", Cryptology ePrint Archive: Report 2002/163, November 2002.

Appendix A. Changes from -01

- o Added combined mutual authentication in section 3.1
- o Changed reference from SP 800-52 to SP 800-57, part 3
- o In section 6.2 changed terminology to tunnel MitM and security policy enforcement
- o Reworded text in section 3.2 to clarify MITM protection
- o Added more specific text about derivation of the CTK
- o Removed resource constrained section
- o Added support for Non EAP authentication as a use for extensibility
- o Added text to emergency services section to emphasize that sensitive information should not be sent if the server is unauthenticated.
- o Reworded TLS requirements
- o Reworded external data protection requirements
- o Added text to section 4.6 that states method must not be re-defined inside the tunnel.
- o Editorial fixes

Appendix B. Changes from -02

- o Editorial Fixes
- o Clarified client authentication during tunnel establishment
- o Changed text so that the tunnel method MUST meet all MUST and SHOULD requirements relevant to EAP methods in RFCs 4962 and 5247

Appendix C. changes from -03

- o Resolution of open issues:
<http://trac.tools.ietf.org/wg/emu/trac/report/9>

- o Revised section 2 to match other similar RFC(Issue 6)
- o Cleaned up section 3.2 (issue 8)
- o Clarified identity protection scope in section 3.4 and 4.2.1.4(issue 9)
- o Changed Emergency Services to anonymous authentication(section 3.5)(issue 10)
- o Clarified section 4.1.1 (issue 15)
- o Cleaned up TLS requirements in section 4.2.1(issue 11)
- o Replaced text in 4.2.1.1.3 with suitable reference
- o Improved wording in 4.2.3 and 6.3 (issue 13)
- o Update internationalization requirements in 4.3.6 and 4.5.2 (Issues 25,18)
- o Updated text in 4.5.1 (issue 16)
- o Changed meta-data to SHOULD in 4.5.3 and 4.6.5(Issue 20)
- o Changed chained methods to SHOULD in 4.6.2(issue 19)
- o Added security consideration for inner method termination(issue 24)
- o Updated references
- o Editorial changes(issues 7,22,17)

Authors' Addresses

Katrin Hoepfer
Motorola, Inc.
1301 E Algonquin Rd
Schaumburg, IL 60196
USA

Email: khoepfer@motorola.com

Stephen Hanna
Juniper Networks
3 Beverly Road
Bedford, MA 01730
USA

Email: shanna@juniper.net

Hao Zhou
Cisco Systems, Inc.
4125 Highlander Parkway
Richfield, OH 44286
USA

Email: hzhou@cisco.com

Joseph Salowey (editor)
Cisco Systems, Inc.
2901 3rd. Ave
Seattle, WA 98121
USA

Email: jsalowey@cisco.com

