               The Tunneled Extensible Authentication Method (TEAM)
                        draft-zorn-emu-team-02

Abstract

   The Extensible Authentication Protocol (EAP) provides support for
   multiple authentication methods.  This document defines the Tunneled
   Extensible Authentication Method (TEAM), which provides an encrypted
   and authenticated tunnel based on transport layer security (TLS) that
   encapsulates EAP authentication mechanisms.  TEAM uses TLS to protect
   against rogue authenticators, protect against various attacks on the
   confidentiality and integrity of the inner EAP method exchange and
   provide EAP peer identity privacy.  TEAM also provides support for
   chaining multiple EAP mechanisms, cryptographic binding between
   authentications performed by inner EAP mechanisms and the tunnel,
   exchange of arbitrary parameters (TLVs), and fragmentation and
   reassembly.

Table of Contents

1.  Introduction

   The Extensible Authentication Protocol (EAP), defined in [RFC3748],
   provides support for multiple authentication methods.  EAP over PPP
   [RFC3748] is typically deployed with leased lines or modem
   connections.  [IEEE.802-1X.2004] defines EAP over IEEE 802 local area
   networks (EAPOL).

   Since its initial development, a number of weaknesses in the EAP
   framework have become apparent.  These include lack of support for:

      Identity protection
      Protected method negotiation
      Protected notification messages
      Protected termination messages
      Sequences of EAP methods
      Fragmentation and reassembly
      Exchange of arbitrary parameters in a secure channel
      Optimized re-authentication

   In addition, some EAP methods lack the following features:

      Mutual authentication
      Resistance to dictionary attacks
      Adequate key generation

   By wrapping the EAP protocol within TLS, TEAM addresses deficiencies
   in EAP or EAP methods.  Benefits of TEAM include:

   Identity protection
      By encrypting the identity exchange, and allowing client identity
      to be provided after negotiation of the TLS channel, TEAM provides
      for identity protection.

   Dictionary attack resistance
      By conducting the EAP conversation within a TLS channel, TEAM
      protects EAP methods that might be subject to an offline
      dictionary attack were they to be conducted in the clear.

   Protected negotiation
      Since within TEAM, the EAP conversation is authenticated,
      integrity and replay protected on a per-packet basis, the EAP
      method negotiation that occurs within TEAM is protected, as are
      error messages sent within the TLS channel (TLS alerts or EAP
      Notification packets).  EAP negotiation outside of TEAM is not
      protected.

Header protection
    Within TEAM, TLS provides per-packet encryption, authentication,
    integrity and replay protection for the EAP conversation.  As a
    result, the Type-Data field within TEAM (including the EAP header
    of the EAP method within TEAM) is protected against modification.
    However, the EAP header of TEAM itself is not protected against
    modification, including the Code, Identifier and Type fields.

Protected termination
    By sending success/failure indications within the TLS channel,
    TEAM provides support for protected termination of the EAP
    conversation.  This prevents an attacker from carrying out denial
    of service attacks by spoofing EAP Failure messages, or fooling
    the EAP peer into accepting a rogue NAS, by spoofing EAP Success
    messages.

Fragmentation and Reassembly
    Since EAP does not include support for fragmentation and
    reassembly, individual methods need to include this capability.
    By including support for fragmentation and reassembly within TEAM,
    methods leveraging TEAM do not need to support this on their own.

Fast reconnect
    Where EAP is used for authentication in wireless networks, the
    authentication latency is a concern.  As a result, it is valuable
    to be able to do a quick re-authentication on roaming between
    access points.  TEAM supports this capability by leveraging the
    TLS session resumption facility, and any EAP method running under
    TEAM can take advantage of it.

Standard key establishment
    In order to provide keying material for a wide range of link layer
    ciphersuites, EAP methods need to provide keying material.  Key
    derivation is complex.  TEAM provides for key establishment by
    relying on the widely implemented and well-reviewed TLS [RFC5246]
    key derivation mechanism.  TEAM provides keying material for any
    EAP method running within it.

Sequencing of multiple EAP methods
    In order to enhance security, TEAM implementations may choose to
    provide multi-factor authentication that validates different
    identities (for example user and machine identities) and/or uses
    different credentials of the same or different identities of the
    peer (e.g. user password and machine cert).  TEAM provides a
    standard way to chain different types of authentication mechanisms
    supporting different types of credentials.

Protected exchange of arbitrary parameters
    Type-Length-Value (TLV) tuples provide a way to exchange arbitrary
    information between peer and EAP server within a secure channel.
    This information can include signaling parameters for the EAP
    protocol, provisioning parameters, media specific and environment
    specific data, and authorization parameters.  The advantage of
    using TEAM TLVs is that every EAP method does not have to be
    modified.

Credential provisioning
    TEAM supports provisioning of certificate trust anchors by the
    server using TLVs and can be extended to support provisioning of
    other peer credentials.

Optimized for light weight devices
    In order to support peers that may not support certificate
    ciphersuites, and may not support provisioning of certificate
    trust anchors, TEAM enables negotiation of other TLS ciphersuites.

Server unauthenticated tunnel provisioning mode
    In some cases, the peer may only support password credentials and
    may not be provisioned with a certificate trust anchor.

    In server unauthenticated tunnel provisioning mode, a TEAM peer
    can authenticate using a password, in order to be provisioned with
    a pre-shared key and other credentials that can be used for
    subsequent authentication.  In server unauthenticated tunnel
    provisioning mode the TEAM peer does not authenticate the server.
    As a result, it is possible for an attacker to act as a man-in-
    the-middle during the initial exchange in order to attack the
    inner exchange.  For this reason, when performing server
    unauthenticated tunnel provisioning mode the inner method MUST be
    resistant to dictionary attack.

2.  Requirements Language

   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.  Terminology

   This document frequently uses the following terms:

Access Point
    A Network Access Server implementing 802.11.

Authenticator
    The end of the link initiating EAP authentication.  This term is
    also used in [IEEE.802-1X.2004]. and has the same meaning in this
    document.

Backend Authentication Server
    A backend authentication server is an entity that provides an
    authentication service to an Authenticator.  When used, this
    server typically executes EAP methods for the Authenticator.  This
    terminology is also used in [IEEE.802-1X.2004].

EAP server
    The entity that terminates the EAP authentication method with the
    peer.  In the case where no backend authentication server is used,
    the EAP server is part of the Authenticator.  In the case where
    the authenticator operates in pass-through mode, the EAP server is
    located on the backend authentication server.

Link layer ciphersuite
    The ciphersuite negotiated for use at the link layer.

NAS
    Short for "Network Access Server".

Peer
    The end of the link that responds to the authenticator.  In
    [IEEE.802-1X.2004], this end is known as the Supplicant.

TLS Ciphersuite
    The ciphersuite negotiated for protection of Phase 2 of the TEAM
    conversation Section 4.4.

EAP Master key (MK)
    A key derived between the TEAM client and server during the
    authentication conversation, and that is kept local to TEAM and
    not exported or made available to a third party.

Master Session Key (MSK)
    Keying material that is derived between the EAP peer and server
    and exported by the EAP method.  The MSK is at least 64 octets in
    length.  In existing implementations, a AAA server acting as an
    EAP server transports the MSK to the authenticator.

Extended Master Session Key (EMSK)
   Additional keying material derived between the EAP client and
   server that is exported by the EAP method.  The EMSK is at least
   64 octets in length.  The EMSK is not shared with the
   authenticator or any other third party.  The EMSK is reserved for
   future uses that are not defined yet.

Type Length Value (TLV)
   The TEAM protocol utilizes objects in Type Length Value (TLV)
   format.  The TLV format is defined in Section 6.1 of this
   document.


4.  Protocol Overview

   TEAM is comprised of a two-part conversation:

   1.  In Phase 1 a TLS session is negotiated, with the server
       authenticating to the client and (optionally) the client to the
       server.  The negotiated key is then used to protect Phase 2 of
       the conversation.

   2.  In Phase 2, within the TLS session, zero or more EAP methods are
       carried out.  Phase 2 completes with a success/failure indication
       protected by the TLS session or a protected error (TLS alert).

   In the following sections, we discuss the TEAM operational model, its
   support for EAP method sequencing and provide an overview of each of
   the parts of the TEAM conversation.

4.1.  Operational Model

   In EAP, the EAP server may be implemented either within a Network
   Access Server (NAS) or on a backend authentication server.  Where the
   EAP server resides on a NAS, the NAS is required to implement the
   desired EAP methods, and therefore needs to be upgraded to support
   each new EAP method.

   One of the goals of EAP is to enable development of new
   authentication methods without requiring deployment of new code on
   the Network Access Server (NAS).  Where a backend authentication
   server is deployed, the NAS acts as a "passthrough" and need not
   understand specific EAP methods.

   This allows new EAP methods to be deployed on the EAP peer and
   backend authentication server, without the need to upgrade code
   residing on the NAS.

Figure 1 illustrates the relationship between the EAP peer, NAS and
EAP server.  As shown in the figure, the EAP conversation occurs
between the EAP peer and EAP server, "passing through" the NAS.  In
order for the conversation to proceed in the case where the NAS and
EAP server reside on separate machines, the NAS and EAP server need
to establish trust beforehand.

```
    +-+-+-+-+-+                      +-+-+-+-+-+
    |         |                      |         |
    | Link    |                      | Link    |
    | Layer   |                      | Layer   |
    | Cipher- |                      | Cipher- |
    | Suite   |                      | Suite   |
    |         |                      |         |
    +-+-+-+-+-+                      +-+-+-+-+-+
        ^                                ^
        |                                |
        |                                |
        |                                |
        V                                V
    +-+-+-+-+-+       EAP            +-+-+-+-+-+  Trust +-+-+-+-+-+
    |         |    EAP              |         |    |<======>|         |
    |         |    Conversation     |         |    |        |         |
    | EAP     |  |<============================>|  EAP     |
    | Peer    |  | (over PPP,      |  NAS    |  |  Server  |
    |         |  | 802.11,etc.)    |         |  |<=======|         |
    |         |                   |         |    Keys   |         |
    |         |                   |         |    |        |         |
    +-+-+-+-+-+                    +-+-+-+-+-+    +-+-+-+-+-+
        ^                                              ^
        |                                              |
        | EAP API                                      | EAP API
        |                                              |
        V                                              V
    +-+-+-+-+-+                              +-+-+-+-+-+
    |         |                              |         |
    |         |                              |         |
    | EAP     |                              | EAP     |
    | Method  |                              | Method  |
    |         |                              |         |
    +-+-+-+-+-+                              +-+-+-+-+-+
```
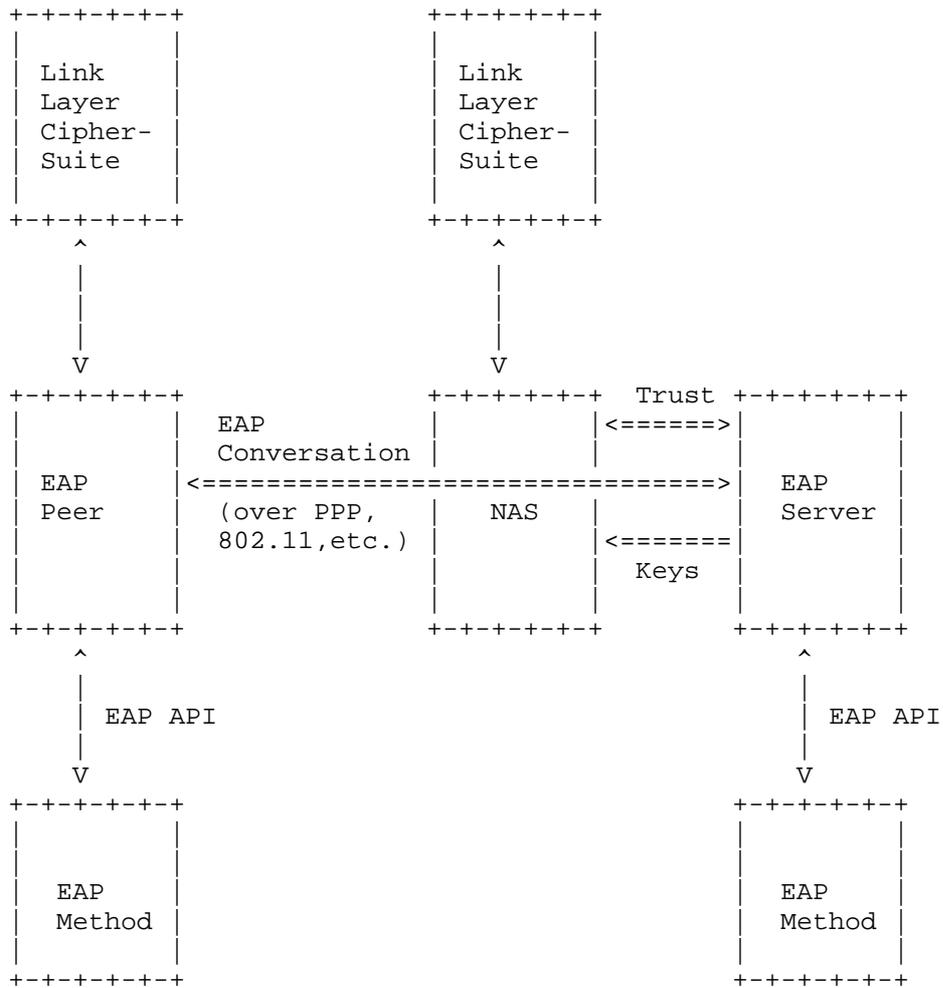
Figure 1: Relationship between EAP client, backend authentication
                        server and NAS

In TEAM, the conversation between the EAP peer and the EAP server is
encrypted, authenticated, integrity and replay protected within a TLS
channel.

As a result, where the NAS acts as a "passthrough" it does not have
knowledge of the TLS master secret derived between the peer and the
EAP server.  In order to provide keying material for link-layer
ciphersuites, the NAS obtains the master session key, which is
derived from a one-way function of the TLS master secret as well as
keying material provided by EAP methods protected within a TLS
channel.  This enables the NAS and EAP peer to subsequently derive
transient session keys suitable for encrypting, authenticating and
integrity protecting session data.  However, the NAS cannot decrypt
the TEAM conversation or spoof session resumption, since this
requires knowledge of the TLS master secret.

4.2.  Sequences

EAP [RFC3748] prohibits use of multiple authentication methods within
a single EAP conversation, except when tunneled methods such as TEAM
are used.  This restriction was imposed in order to limit
vulnerabilities to man-in-the-middle attacks as well as to ensure
compatibility with existing EAP implementations.

Within TEAM these concerns are addressed since TEAM includes support
for cryptographic binding to address man-in-the-middle attacks, as
well as version negotiation so as to enable backward compatibility
with future versions of the protocol.

Within this document, the term "sequence" refers to a series of EAP
authentication methods run in sequence or TLV exchanges before or
after EAP methods.  The methods need not be distinct - for example,
EAP-TLS could be run initially with machine credentials followed by
the same protocol authenticating with user credentials.

TEAM supports initiating additional EAP method(s) after a successful
or a failed EAP method.  The result of failure of a EAP method does
not always imply a failure of the overall authentication.  The
overall result of authentication depends on the policy at EAP server
and the peer.  For example, successful authentication might require a
successful machine authentication followed by a successful user
authentication.  Alternatively, if machine authentication fails, then
user authentication can be attempted.  TEAM does not support
initiating multiple EAP methods simultaneously.

4.3.  TEAM Phase 1

4.3.1.  Initial identity exchange

The TEAM conversation typically begins with an optional identity
exchange.  The authenticator will typically send an EAP-Request/
Identity packet to the peer, and the peer will respond with an EAP-

Response/Identity packet to the authenticator.

The initial identity exchange is used primarily to route the EAP
conversation to the EAP server.  Since the initial identity exchange
is in the clear, the peer MAY decide to place a routing realm instead
of its real name in the EAP-Response/Identity.  The real identity of
the peer can be established later, during Phase 2.

If the EAP server is known in advance (such as when all users
authenticate against the same backend server infrastructure and
roaming is not supported), or if the identity is otherwise determined
(such as from the dialing phone number or client MAC address), then
the Identity exchange MAY be omitted.

Once the optional initial Identity Request/Response exchange is
completed, while nominally the EAP conversation occurs between the
authenticator and the peer, the authenticator MAY act as a
passthrough device, with the EAP packets received from the peer being
encapsulated for transmission to a backend authentication server.
However, TEAM does not require a backend authentication server; if
the authenticator implements TEAM, then it can authenticate local
users.

In the discussion that follows, we will use the term "EAP server" to
denote the ultimate endpoint conversing with the peer.

4.3.2.  TLS Session Establishment

In this section, the protocol is described.  While this section will
often describe negotiation of a certificate-based ciphersuite within
TLS, TEAM supports negotiation of other ciphersuites (for example,
ciphersuites that do not use certificates) or extensions.  However,
the conversation may slightly differ if other TLS ciphersuites or
extensions are used.

Once having received the peer's identity, and determined that TEAM
authentication is to occur, the EAP server MUST respond with a TEAM/
Start packet, which is an EAP-Request packet with EAP-Type=TEAM, the
Start (S) bit set, the TEAM version as specified in Section 4.3.4,
and optionally, the Server-Identifier TLV (Section 6.14).

Assuming that the peer supports TEAM, the TEAM conversation will then
begin, with the peer sending an EAP-Response packet with EAP-
Type=TEAM.  The Type-Data field of the EAP-Response Packet will
encapsulate one or more TLS records containing the TLS handshake
messages.  As defined in [RFC5246], the TLS handshake is used to
negotiate parameters and cryptographic keys and may take several
roundtrips between the TLS client and server.

The version offered by the TLS client and server MUST be TLS v1.0 or later.  TEAM implementations need not necessarily support all TLS ciphersuites listed in [RFC5246].  Not all TLS ciphersuites are supported by available TLS tool kits and licenses may be required in some cases.

To ensure interoperability, TEAM peers and servers MUST support the TLS v1.1 [RFC5246] mandatory-to-implement ciphersuite:

    TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA

In addition, TEAM servers SHOULD support and be able to negotiate all of the following TLS ciphersuites:

    TLS_RSA_WITH_3DES_EDE_CBC_SHA
    TLS_RSA_WITH_RC4_128_MD5
    TLS_RSA_WITH_RC4_128_SHA
    TLS_RSA_WITH_AES_128_CBC_SHA

In addition, TEAM peers SHOULD support at least one of the following TLS ciphersuites:

    TLS_RSA_WITH_3DES_EDE_CBC_SHA
    TLS_RSA_WITH_RC4_128_MD5
    TLS_RSA_WITH_RC4_128_SHA
    TLS_RSA_WITH_AES_128_CBC_SHA

TLS as described in [RFC5246] supports compression as well as ciphersuite negotiation.  Therefore during the TEAM Phase 1 conversation the TEAM endpoints MAY request or negotiate TLS compression.

If the full TLS handshake is performed, then the first payload of TEAM Phase 2 MAY be sent along with finished handshake message to reduce number of round trips.

Since after the TLS session is established, another complete EAP negotiation will occur and the peer will authenticate using a secondary mechanism, with TEAM the client need not authenticate as part of TLS session establishment.

Note that since TLS client certificates are sent in the clear, if identity protection is required, then it is possible for the TLS authentication to be re-negotiated after the first server authentication.  Alternatively, if identity protection is required, then it is possible to perform certificate authentication using a EAP method (for example: EAP-TLS [RFC5216]) within the TLS session during TEAM Phase 2.

To accomplish this, the server will typically not request a
certificate in the server_hello, then after the server_finished
message is sent, and before TEAM Phase 2 begins, the server MAY send
a TLS hello_request.  This allows the client to perform client
authentication by sending a client_hello if it wants to, or send a
no_renegotiation alert to the server indicating that it wants to
continue with TEAM Phase 2 instead.  Assuming that the client permits
renegotiation by sending a client_hello, then the server will respond
with server_hello, a certificate and certificate_request messages.
The client replies with certificate, client_key_exchange and
certificate_verify messages.  Since this re-negotiation occurs within
the encrypted TLS channel, it does not reveal client certificate
details.

4.3.3.  Session Resumption

   The purpose of the sessionId within the TLS protocol and the Server-
   Identifier TLV in TEAM is to allow for improved efficiency in the
   case where a client repeatedly attempts to authenticate to an EAP
   server within a short period of time.  This capability is
   particularly useful for support of wireless roaming.

   In order to help the peer choose a sessionID that belongs to the
   specific server, the EAP server MAY send an identifier (Server-
   Identifier TLV) that the peer can use as a hint.  The Server-
   Identifier TLV MAY be sent in the first TEAM packet from the EAP
   server to the peer.  In order to detect modification of the Server-
   Identifier TLV, the Server-Identifier TLV is included in calculation
   of the compound MAC.

   It is left up to the peer whether to attempt to continue a previous
   session, thus shortening the TEAM Phase 1 conversation.  Typically
   the peer's decision will be made based on the time elapsed since the
   previous authentication attempt to that EAP server.

   Based on the sessionId chosen by the peer, and the time elapsed since
   the previous authentication, the EAP server will decide whether to
   allow the continuation, or whether to choose a new session.

   If the EAP server is resuming a previously established session, then
   it MUST include only a TLS change_cipher_spec message and a TLS
   finished handshake message after the server_hello message.  The
   finished message contains the EAP server's authentication response to
   the peer.

   If the preceding server_hello message sent by the EAP server in the
   preceding EAP-Request packet indicated the resumption of a previous
   session, then the peer MUST send only the change_cipher_spec and

finished handshake messages.  The finished message contains the
peer's authentication response to the EAP server.  The latter
contains the EAP server's authentication response to the peer.  The
peer will verify the hash in order to authenticate the EAP server.

If authentication fails, then the peer and EAP server MUST follow the
error handling behavior specified in Section 4.5

Even if the session is successfully resumed with the same EAP server,
the peer and EAP server MUST NOT assume that either will skip inner
EAP methods.  The peer may have roamed to a network which may use the
same EAP server, but may require conformance with a different
authentication policy, and therefore may require inner EAP
authentication methods.

4.3.4.  Version Negotiation

   TEAM packets contain a three bit version field, which enables TEAM
   implementations to be backward compatible with previous versions of
   the protocol.  This specification documents version1 of the TEAM
   protocol; implementations of this specification MUST use a version
   field set to 1.  Version negotiation proceeds as follows:

   1.  In the first EAP-Request sent with EAP-Type=TEAM, the EAP server
       MUST set the version field to the highest supported version
       number.

   2.  If the EAP peer supports that version of the protocol, it MUST
       respond with an EAP-Response of EAP-Type=TEAM, and the version
       number proposed by the EAP server.

   3.  If the EAP peer does not support that version, it responds with
       an EAP-Response of EAP-Type=TEAM and the highest supported
       version number.

   4.  If the TEAM server does not support the version number proposed
       by the TEAM peer, it either starts a different EAP type or
       terminates the conversation by sending an EAP-Failure, depending
       on the server policy.

   The version negotiation procedure guarantees that the EAP peer and
   server will agree to the latest version supported by both parties.
   If version negotiation fails, then use of TEAM will not be possible,
   and another mutually acceptable EAP method will need to be negotiated
   if authentication is to proceed.

   The TEAM version field is not protected by TLS and therefore can be
   modified in transit.  In order to detect modification of the TEAM

version which could occur as part of a "downgrade" attack, the peer
and EAP server check if the version it sent during negotiation is
same as the version claimed to be received by the other party.  Each
party uses the Crypto-Binding TLV (Section 6.5) to inform the other
party of the version number it received during the TEAM version
negotiation.  The receiver of the Crypto-Binding TLV must verify that
the version in the Crypto-Binding TLV matches the version it sent
during TEAM version negotiation.

4.4.  TEAM Phase 2

   The second part of the TEAM conversation typically consists of a
   complete EAP conversation occurring within the TLS session negotiated
   in TEAM Phase 1, ending with protected termination using the Result
   TLV.  TEAM Phase 2 will occur only if establishment of a new TLS
   session in Phase 1 is successful or a TLS session is successfully
   resumed in Phase 1.  In cases where a new TLS session is established
   in TEAM Phase 1, the first payload of the Phase 2 conversation MAY be
   sent by the EAP server along with the finished message to save a
   round-trip.

   Phase 2 SHOULD NOT occur if the EAP Server authenticates
   unsuccessfully, and MUST NOT occur if establishment of the TLS
   session in Phase 1 was not successful or a TLS fatal error has been
   sent terminating the conversation.

   Since all packets sent within the TEAM Phase 2 conversation occur
   after TLS session establishment, they are protected using the
   negotiated TLS ciphersuite.  All EAP packets of the EAP conversation
   in Phase 2 including the EAP header of the inner EAP method are
   protected using the negotiated TLS ciphersuite.

   Phase 2 may not always include a EAP conversation within the TLS
   session, referred to in this document as inner EAP methods.  However,
   Phase 2 MUST always end with either protected termination or
   protected error termination (e.g.  TLS alert).

   Within Phase 2, protected EAP conversation and protected termination
   packets are always carried within TLVs.  There are TLVs defined for
   specific purposes such as carrying EAP-authentication messages and
   carrying cryptographic binding information.  New TLVs may be
   developed for other purposes.

4.4.1.  Protected Conversation

   Phase 2 of the TEAM conversation typically begins with the EAP server
   sending an optional EAP-Request/Identity packet to the peer,
   protected by the TLS ciphersuite negotiated in Phase 1 of TEAM.  The

peer responds with an EAP-Response/Identity packet to the EAP server, containing the peer's userId.  Since this Identity Request/Response exchange is protected by the ciphersuite negotiated in TLS, it is not vulnerable to snooping or packet modification attacks.

After the TLS session-protected Identity exchange, the EAP server will then select authentication method(s) for the peer, and will send an EAP-Request with the Type field set to the initial method.  As described in [RFC3748], the peer can NAK the suggested EAP method, suggesting an alternative.  Since the NAK will be sent within the TLS channel, it is protected from snooping or packet modification.  As a result, an attacker snooping on the exchange will be unable to inject NAKs in order to "negotiate down" the authentication method.  An attacker will also not be able to determine which EAP method was negotiated.

The EAP conversation within the TLS protected session may involve a sequence of zero or more EAP authentication methods; it completes with the protected termination described in Section 4.4.2 Several TLVs may be included in each Request and Response.  EAP packets are always encapsulated within EAP Payload TLVs.

In a typical EAP conversation, the result of the conversation is communicated by sending EAP Success or EAP Failure packets after the EAP method is complete.  The EAP Success or Failure packet is considered the last packet of the EAP conversation; and therefore cannot be used when sequences need to be supported.  Hence, instead of using the EAP Success or EAP Failure packet, both peer and EAP server MUST use the Intermediate-Result TLV (Section 6.10) to communicate the result.

In a typical EAP conversation, the EAP Success or EAP Failure is considered the last packet of the EAP conversation.  Within TEAM, the EAP server can start another EAP method after success or failure of the previous EAP method inside the protected session.

In a sequence of more than one EAP authentication method, to make sure the same parties are involved in tunnel establishment and successful completion of previous inner EAP methods, before completing negotiation of the next EAP method, both peer and EAP server MUST use cryptographic binding (Crypto-Binding TLV Section 6.5).

The Intermediate-Result TLV is used to indicate the result of a individual successful EAP method, and the Result TLV (Section 6.2) is used to indicate result of the entire TEAM conversation.

The Intermediate-Result and Crypto-Binding TLVs MUST be sent after

each EAP method that was successful.  If the EAP method failed, or if
the EAP method negotiation did not complete, then an Intermediate-
Result TLV MAY be included, and the Crypto-Binding TLV MUST NOT be
included.  An exception is that the Crypto-Binding TLV MUST be sent
along with a protected success/failure indication (see
Section 4.4.2).

If these TLVs are not sent after a successful EAP method, it should
be considered a tunnel compromise error by peer and EAP server,
resulting in the termination of the conversation (as described in
Section 4.5).

A subsequent EAP conversation can be started after both TLVs are
exchanged in a TLV packet.  Alternatively, if a subsequent EAP
conversation is being attempted, then in order to reduce round trips,
both TLVs SHOULD be sent with the EAP-Payload of the first EAP packet
of the next EAP conversation (for example, EAP-Identity or EAP packet
of the EAP method).  Alternatively, if the next packet is the
protected success/failure packet, then in order to reduce round
trips, both TLVs MUST be sent with the protected success/failure
packet.

If the EAP server sends a valid Crypto-Binding TLV to the peer, the
peer MUST respond with a Crypto-Binding TLV.  If the Crypto-Binding
TLV is invalid, it should be considered a tunnel compromise error by
the peer.  If the peer does not respond with a TLV packet containing
the Crypto-Binding TLV, it MUST be considered a tunnel compromise
error by the EAP server.

Within a TEAM part 2 conversation, a peer MAY request the trusted
root of a server certificate using a Server-Trusted-Root TLV
(Section 6.16), and the EAP server MAY respond with a Server-Trusted-
Root TLV to the peer.  The Server-Trusted-Root TLV can be exchanged
in regular authentication mode or server unauthenticated tunnel
provisioning mode.

After the peer has determined that it has successfully authenticated
the EAP server and determined that the tunnel and inner EAP methods
were between the same peer and EAP server by validating the Crypto-
Binding TLV, it MAY send one or more Server-Trusted-Root TLVs (marked
as optional) to request the trusted root of server certificate from
the EAP server.  The peer may receive a response, but is not required
to use the trusted root received from the EAP server.

If the EAP server has determined that it has successfully
authenticated the peer and determined that the tunnel and inner EAP
methods were between the same peer and EAP server by validating the
Crypto-Binding TLV, then it MAY respond with the the server-trusted-

root containing the PCKS#7 TLV (Section 6.17).

4.4.2.  Protected Termination

Phase 2 of the TEAM conversation is completed by the exchange of
success/failure indications (Result TLV) within a TLV packet
protected by the TLS session.

Even if Crypto-Binding TLVs have been exchanged in previous
conversations, the Crypto-Binding TLV MUST be included in both
protected success/failure (Result TLV) indications.  If the TLVs are
not included, or if the TLVs are invalid, it should be considered a
tunnel compromise error, and the peer and EAP server MUST follow the
rules described in Section 4.5 to abort the conversation.

The Result TLV is sent within the TLS channel.  The TEAM client then
replies with a Result TLV.  The conversation concludes with the TEAM
server sending a cleartext success/failure indication.

The only outcome which should be considered as successful
authentication is when a Result TLV of Status=Success is answered by
the peer with a Result TLV of Status=Success.

The combinations (Result TLV=Failure, Result TLV=Success), (Result
TLV=Failure, Result TLV=Failure), (no TLVs exchange or no protected
success or failure) should be considered an authentication failure by
both the peer and EAP server.  Once the peer and EAP server consider
that authentiation has failed, these are the last packets inside the
protected tunnel.  These combinations are considered an
authentication failure regardless of whether a cleartext EAP Success
or EAP Failure packet is subsequently sent.

If the EAP server wants authentication to fail, it sends the TLV
response with Result TLV=Failure.  If the EAP server sends a failure,
the peer MUST respond with Result TLV=Failure and the Crypto-Binding
TLV, without any other mandatory TLVs.  The Crypto-Binding TLV is
calculated using the key derivation formula in Section 2.5; if for
some reason one or more inner EAP method MSKs were not derived, then
these MSKs are assumed to be null.

If the EAP server has sent the success indication (Result
TLV=Success), the peer is allowed to refuse to accept a Success
message from the EAP server since the client's policy may require
completion of certain EAP methods or the client may require
credentials.

If the EAP server has sent a success indication (Result TLV=success),
and the peer wants authentication to fail, it sends the TLV response

with Result TLV=Failure and Crypto-Binding TLV.

After the EAP server returns success, if the peer wants to request
the EAP server to continue conversation, it sends a Result
TLV=Success along with a Request-Action TLV with the appropriate
action (e.g.  Negotiate-EAP, or Process-TLV).  If the Request-Action
TLV is set to mandatory, then the EAP server MUST process the action,
or return status=failure, closing the conversation inside the tunnel.
If the Request-Action TLV is set to optional, then the EAP server can
ignore the TLV and return Result TLV=Success again, closing the
conversation inside the tunnel.

4.4.3.  Provisioning of Credentials

TEAM supports built-in provisioning of certificate trust anchors and
can be extended to provisioning of other types of credentials.  The
following two provisioning modes are suported:

1.  Provisioning inside a server authenticated TLS tunnel
2.  Provisioning inside a server unauthenticated TLS tunnel

4.4.3.1.  Provisioning Inside a Server Authenticated TLS Tunnel

After regular authentication in TEAM Phase 2, the peer and EAP server
can use the Server-Trusted-Root TLV to request and provision peer
credentials.  The provisioning payload is exchanged after the peer
and EAP server have determined that both have successfully
authenticated each other (either thru TLS handshake and/or inner EAP
method), and the tunnel and inner EAP methods are between the same
peers.

After the peer has determined that it has successfully authenticated
the EAP server and determined that the tunnel and inner EAP methods
were between the same peer and EAP server by validating the Crypto-
Binding TLV, it MAY send one or more Server- Trusted-Root TLVs
(marked as optional) to request credentials from the EAP server.  The
EAP server will send corresponding credentials in the Server-Trusted-
Root TLVs if its internal policy has been satisfied.  It may ignore
the credential provisioning request or request additional
authentication methods if its policy so dictates.  The peer may
receive a credential, but is not required to use the credentials
received from the EAP server.

4.4.3.2.  Provisioning Inside a Server Unauthenticated TLS Tunnel

In some cases, the peer may lack the credentials necessary to
authenticate the server in the TLS handshake.  At the same time,
bootstrapping the information to the peer out of band may be

prohibitive from a deployment cost perspective.  It can rely on the
inner EAP method using existing credentials to authenticate the
server.

In this provisioning mode, as part of TEAM Phase 1, if the peer does
not authenticate, or does not successfully authenticate the EAP
server during TLS negotiation, it can decide to go into server
unauthenticated tunnel provisioning mode.  In a certificate-based TLS
handshake, the peer verifies that the EAP server possesses the
private key corresponding to the public key contained in the
certificate presented by the EAP server.  However, the peer does not
verify whether the certificate presented by the server chains to a
provisioned trust anchor, as the peer may not be configured with a
certificate trust anchor required to validate the server certificate.
If the peer cannot verify that the server possesses the corresponding
private key, or if the certificate presented by the server is
unacceptable for any reason other than the lack of an appropriate
trust anchor, the peer MUST NOT use this provisioning mode.  Assuming
that the server demonstrates possession of the private key, the peer
continues with establishment of the tunnel (TEAM Phase 2).  In a
certificate-less TLS handshake the peer and server perform an
anonymous exchange.  There is no attempt by the peer to verify the
server's identity.  In both the certificate-based and certificate-
less TEAM Phase 1 exchange for the Server Unauthenticated mode, it is
possible that the TLS channel (TEAM Phase 2) may be terminated by an
attacker.  For this reason the TEAM Phase 2 exchange MUST be
resistant to dictionary attack.

The TEAM Phase 2 conversation is unchanged in this mode, except that
the peer will only accept an EAP method supporting mutual
authentication, key derivation and resistance to dictionary attack
that is compatible with its initial credentials (such as EAP-pwd
[RFC5931]).  The peer then uses the Crypto-Binding TLV to validate
that the same server terminates both the TLS channel and the
successfully completed EAP method, thereby verifying that the
exchange was not subject to a man-in-the-middle attack.  Assuming
that the Crypto- Binding TLV exchange is successful, the peer will
request and the server will subsequently provide a trusted root,
using the Server-Trusted-Root TLV.

Once the initial provisioning exchange completes, the peer is
expected to use the provisioned credentials in subsequent TEAM
authentications, and SHOULD NOT continue to use this provisioning
mode.

TEAM servers and peers implementing this provisioning mode MUST
support EAP-pwd [RFC5931] as a TEAM Phase 2 conversation.

TEAM servers implementing this provisioning mode MUST support the
following additional ciphersuites, beyond those specified in
Section 4.3.2:

    TLS_DH_anon_WITH_AES_128_CBC_SHA

TEAM peers implementing this provisioning mode MAY support the
following additional ciphersuites, beyond those specified in
Section 4.3.2:

    TLS_DH_anon_WITH_AES_128_CBC_SHA

4.5.  Error Handling

TEAM does not have its own error message capabilities since:

1.  TLS errors are communicated via TLS alert messages.
2.  Errors within the EAP conversation in TEAM Phase 2 are expected
    to be handled by individual EAP methods.
3.  Violation of the TLV rules (Section 6.19) for inner-TLVs are
    handled using Result-TLVs together with the Error-Code TLV.

If an error occurs at any point in the TLS layer, the EAP server
SHOULD send a TLS alert message instead of the next EAP-request
packet to the peer.  The EAP server SHOULD send an EAP-Request packet
with EAP-Type=TEAM, encapsulating a TLS record containing the
appropriate TLS alert message.  The EAP server SHOULD send a TLS
alert message rather than immediately terminating the conversation so
as to allow the peer to inform the user of the cause of the failure
and possibly allow for a restart of the conversation.  To ensure that
the peer receives the TLS alert message, the EAP server MUST wait for
the peer to reply with an EAP-Response packet.

The EAP-Response packet sent by the peer MAY encapsulate a TLS
client_hello handshake message, in which case the EAP server MAY
allow the TEAM conversation to be restarted, or it MAY contain an
EAP-Response packet with EAP-Type=TEAM and no data, in which case the
TEAM server MUST send an EAP-Failure packet, and terminate the
conversation.

It is up to the EAP server whether to allow restarts, and if so, how
many times the conversation can be restarted.  An EAP server
implementing restart capability SHOULD impose a limit on the number
of restarts, so as to protect against denial of service attacks.

If an error occurs at any point in the TLS layer, the peer SHOULD
send a TLS alert message instead of the next EAP-response packet to
the EAP server.  The peer SHOULD send an EAP-Response packet with

EAP-Type=TEAM, encapsulating a TLS record containing the appropriate
TLS alert message.  The EAP server may restart the conversation by
sending a EAP-Request packet encapsulating the TLS
hello_request_handshake message, in which case the peer MAY allow the
TEAM conversation to be restarted; or the EAP server can response
with EAP Failure.

Any time the peer or the EAP server finds an error when processing
the sequence of exchanges, such as a violation of the TLV rules
Section 6.19, it should send a Result TLV of failure and Error-Code
TLV=Unexpected_TLVs_Exchanged (a Fatal error), and terminate the
tunnel.  This is usually due to an implementation problem and is
considered an fatal error.  The party that receives the Error-Code
TLV=Unexpected_TLVs_Exchanged should terminate the tunnel.

If a tunnel compromise error (see (Section 4.4)) is detected by the
Peer or EAP server, the party SHOULD send a Result TLV of failure
without a Crypto-Binding TLV, and Error-Code TLV=Tunnel-compromise-
error (a Fatal error), and terminate the tunnel.  The party that
receives the Error-Code TLV=Tunnel-compromise error should terminate
the tunnel.

4.6.  Fragmentation and Reassembly

A single TLS record may be up to 16384 octets in length, but a TLS
message may span multiple TLS records, and a TLS certificate message
may in principle be as long as 16MB.

The group of TEAM messages sent in a single round may thus be larger
than the PPP MTU size, the maximum RADIUS packet size of 4096 octets,
or even the Multilink Maximum Received Reconstructed Unit (MRRU).

As described in [RFC1990], the multilink MRRU is negotiated via the
Multilink MRRU LCP option, which includes an MRRU length field of two
octets, and thus can support MRRUs as large as 64 KB.

However, note that in order to protect against reassembly lockup and
denial of service attacks, it may be desirable for an implementation
to set a maximum size for one such group of TLS messages.  Since a
typical certificate chain is rarely longer than a few thousand
octets, and no other field is likely to be anywhere near as long, a
reasonable choice of maximum acceptable message length might be 64
KB.

If this value is chosen, then fragmentation can be handled via the
multilink PPP fragmentation mechanisms described in [RFC1990]. this
is desirable, EAP methods are used in other applications such as
[IEEE.802-11.2007] and there may be cases in which multilink or the

MRRU LCP option cannot be negotiated.  As a result, a TEAM
implementation MUST provide its own support for fragmentation and
reassembly.

Since EAP is an ACK-NAK protocol, fragmentation support can be added
in a simple manner.  In EAP, fragments that are lost or damaged in
transit will be retransmitted, and since sequencing information is
provided by the Identifier field in EAP, there is no need for a
fragment offset field as is provided in IPv4.

TEAM fragmentation support is provided through addition of flag bits
within the EAP-Response and EAP-Request packets, as well as a TLV
Message Length field of four octets.  Flags include the Length
included (L), More fragments (M), and TEAM Start (S) bits.  The L
flag is set to indicate the presence of the four octet TLV Message
Length field, and MUST be set only for the first fragment of a
fragmented TLV message or set of messages.

The TLV Message Length field in the TEAM header is not protected, and
hence can be modified by a attacker.  The TLS record length in the
TLS data is protected.  Hence, if the TLV Message length received in
the first packet (with L bit set) is greater or less than the total
size of TLS messages received including multiple fragments, then the
TLV message length should be ignored.

In order to protect against reassembly lockup and denial of service
attacks, it may be desirable for an implementation to set a maximum
size for a single group of Outer-TLV messages.  Since a typical
certificate chain is rarely longer than a few thousand octets, and no
other field is likely to be anywhere near as long, a reasonable
choice of maximum acceptable message length for all the Outer-TLVs in
a group of messages might be 64 KB.

The M flag is set on all but the last fragment.  The S flag is set
only within the TEAM start message sent from the EAP server to the
peer.  The TLV Message Length field is four octets, and provides the
total length of the TLV message or set of messages that is being
fragmented; this simplifies buffer allocation.

When a peer receives an EAP-Request packet with the M bit set, it
MUST respond with an EAP-Response with EAP-Type=TEAM and no data.
This serves as a fragment ACK.  The EAP server MUST wait until it
receives the EAP-Response before sending another fragment.  In order
to prevent errors in processing of fragments, the EAP server MUST
increment the Identifier field for each fragment contained within an
EAP-Request, and the peer MUST include this Identifier value in the
fragment ACK contained within the EAP-Response.  Retransmitted
fragments will contain the same Identifier value.

Similarly, when the EAP server receives an EAP-Response with the M
bit set, it MUST respond with an EAP-Request with EAP-Type=TEAM and
no TLS data.  This serves as a fragment ACK.  The EAP peer MUST wait
until it receives the EAP-Request before sending another fragment.
In order to prevent errors in the processing of fragments, the EAP
server MUST increment the Identifier value for each fragment ACK
contained within an EAP-Request, and the peer MUST include this
Identifier value in the subsequent fragment contained within an EAP-
Response.

4.7.  Key Derivation

Since the normal TLS keys are used in the handshake, and therefore
should not be used in a different context, new keys must be derived
from the TLS master secret to protect the conversation within the
TEAM tunnel.

Instead of deriving keys specific to link layer ciphersuites, EAP
methods provide a Master Session Key (MSK) used to derive keys in a
link layer specific manner.  The method used to extract ciphering
keys from the MSK is beyond the scope of this document.

TEAM also derives an Extended Master Session Key (EMSK) which is
reserved for use in deriving keys in other ciphering applications.
This draft also does not discuss the format of the attributes used to
communicate the master session keys from the backend authentication
server to the NAS; examples of such attributes are provided in
[RFC2548].

TEAM combines key material from the TLS exchange with key material
from inner key generating EAP methods to provide stronger keys and to
bind inner authentication mechanisms to the TLS tunnel.  Both the
peer and EAP server MUST derive compound MAC and compound session
keys using the procedure described below.

The input for the cryptographic binding includes the following:

1.  The TEAM tunnel key (TK) is calculated using the first 40 octets
    of the (secret) key material generated as described in the EAP-
    TLS algorithm (see Section 2.3 of [RFC5216]) More explicitly, the
    TK is the first 40 octets of the PRF as defined in RFC 5216:

    Key_Material = TLS-PRF-128(  master_secret, "client EAP
                                 encryption", client.random ||
                                 server.random )

2.  The first 32 octets of the MSK provided by each successful inner
    EAP method for each successful EAP method completed within the
    tunnel.

    ISK1..ISKn are the MSK portion of the EAP keying material
    obtained from methods 1 to n.  The ISKj SHALL be the first 32
    octets of the generated MSK of the jth EAP method.  If the MSK
    length is less than 32 octets, it SHALL be padded with 0x00's to
    ensure the MSK is 32 octets.  Similarly, if no keying material is
    provided for the EAP method, then ISKj SHALL be set to zero (e.g.
    32 octets of 0x00).

The key derivation process is based on "extract-then-expand"
technique of HKDF [RFC5869].  Entropy from the TEAM Tunnel Key, TK,
is first extracted into a pseudo-random key and then expanded into a
series of intermediate combined keys, IPMK1..IPMKn, and Compound MAC
keys, CMK1..CMKn.

```
IPMK0 = HKDF-Extract(salt, TK)
for j = 1 to n do
  IPMKj | CMKj = HKDF-Expand(IPMK(j-1),
                             "Inner Methods Compound Keys" | ISKj,
                             60)
done
```

Where:
o  salt is 32 octets of 0x00
o  IPMKj are 40 octets in length
o  CMKj are 20 octets in length used to generate the intermediate
   Compound MACs
and
o  n = the last successful EAP method inside the tunnel at the point
   where the Compound Session Key is derived

4.7.1.  Compound Session Key Derivation

The compound session key (CSK) is derived on both the peer and EAP
server:

    CSK = HKDF-Expand(IPMKn, "Session Key Generating Function", 128)

The length of the CSK MUST be 128 octets.  The first 64 octets SHALL
be taken as the MSK and the second 64 octets SHALL be taken as the
EMSK.  The MSK and EMSK are described in [RFC3748].

4.8.  Ciphersuite Negotiation

   Since TLS supports TLS ciphersuite negotiation, peers completing the
   TLS negotiation will also have selected a TLS ciphersuite, which
   includes key strength, encryption and hashing methods.  However,
   unlike in [RFC5216], within TEAM, the negotiated TLS ciphersuite
   relates only to the mechanism by which the TEAM Phase 2 conversation
   will be protected, and has no relationship to link layer security
   mechanisms negotiated within the PPP Encryption Control Protocol
   (ECP) [RFC1968] or within IEEE 802.11 [IEEE.802-11.2007].

   As a result, this specification currently does not support secure
   negotiation of link layer ciphersuites.

5.  TEAM Protocol Description

5.1.  TEAM Protocol Layers

   TEAM packets may include TLVs both inside and outside the TLS tunnel.
   The term "Outer TLVs" is used to refer to optional TLVs outside the
   TLS tunnel, which are only allowed in the first two messages in the
   TEAM protocol, i.e., the first EAP server to peer message and first
   peer to EAP server message.  If the message is fragmented, the whole
   set of messages is counted as one message.  The term "Inner TLVs" is
   used to refer to TLVs sent within the TLS tunnel.

   In TEAM Phase 1, Outer TLVs are used to help establishing the TLS
   tunnel, but no Inner TLVs are used.  Therefore the layering of TEAM
   Phase 1 is as follows:

```
         +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
         |         TLS       | Optional Outer TLVs       |
         +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
         |                   TEAM                        |
         +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
         |                   EAP                         |
         +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
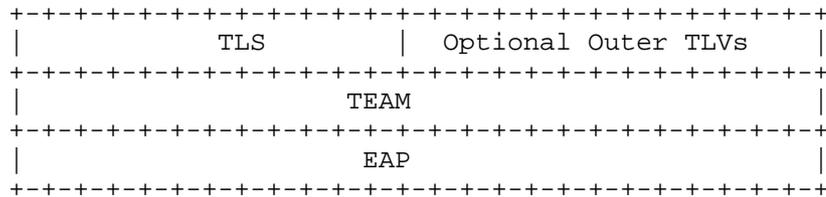
   In Phase 2 of the TEAM conversation, TLS records may encapsulate zero
   or more Inner TLVs, but no Outer TLVs.  EAP packets (including EAP
   header fields) used within tunneled EAP authentication methods are
   carried within Inner TLVs.  Therefore the layering of TEAM Phase 2 is
   as follows:

```
           +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
           |                     EAP                       |
           +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
           |         Inner-TLVs (EAP-Payload TLV)          |
           +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
           |                     TLS                       |
           +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
           |                     TEAM                      |
           +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
           |                     EAP                       |
           +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
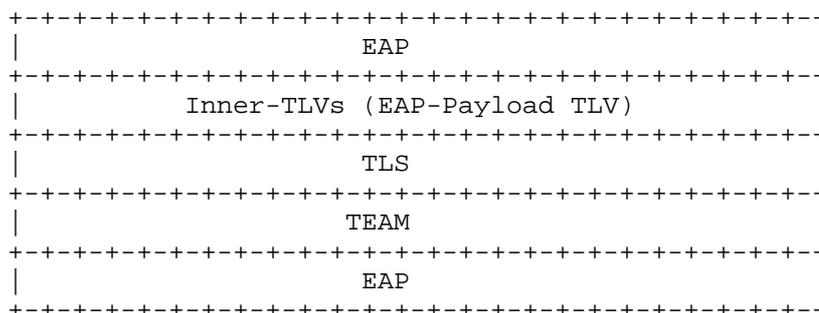
5.2.  TEAM Packet Format

   A summary of the TEAM packet format is shown below.  The fields are
   transmitted from left to right.

```
        0                   1                   2                   3
        0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |     Code      |   Identifier  |            Length             |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |     Type      |  Flags | Ver  |     Fragment Message Length   |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |   Fragment Message Length     |     TLS Message Length        |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |     TLS Message Length        |        TLS Data...            |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |            Outer TLVs...
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
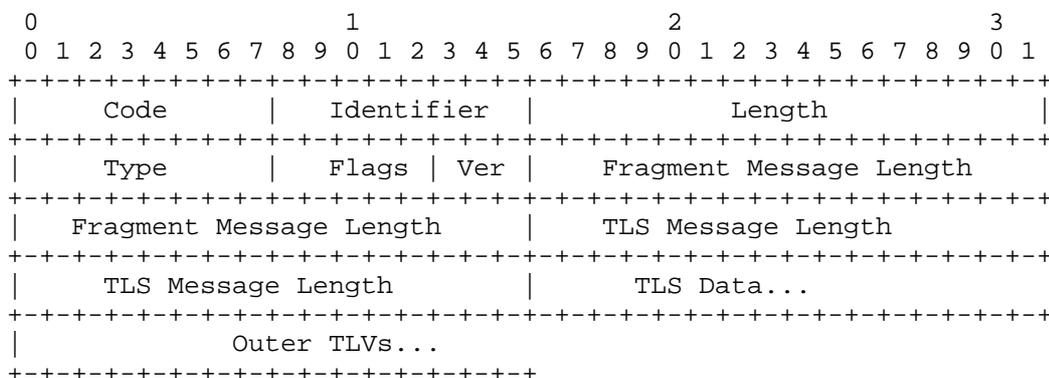
   Code

      1 -  Request
      2 -  Response

   Identifier

      The Identifier field is one octet and aids in matching responses
      with requests.  The Identifier field MUST be changed on each
      Request packet.  The Identifier field in a Response packet MUST
      match the Identifier field from the corresponding Request.

   Length

      The Length field is two octets and indicates the length of the EAP
      packet including the Code, Identifier, Length, Type, Flags,
      Version, Fragmented Length, TLS Message Length, TLS Data, and

Outer-TLV fields.  Octets outside the range of the Length field should be treated as Data Link Layer padding and should be ignored on reception.

Type

   <TBD> - TEAM

Flags

```
 0 1 2 3 4
+-+-+-+-+-+
|L M S T R|
+-+-+-+-+-+
```

   L =  Length included
   M =  More fragments
   S =  TEAM start
   T =  TLS length included
   R =  Reserved (must be zero)

The L bit (Fragmented Message Length included) is set to indicate the presence of the four octet Fragmented Message Length field, and MUST be set for the first fragment of a fragmented TEAM message or set of messages.  The M bit (more fragments) is set on all but the last fragment.  The S bit (TEAM start) is set in a TEAM Start message.  This differentiates the TEAM Start message from a fragment acknowledgment.  The T bit (TLS Message Length included) is set to indicate the presence of the four octet TLS Message Length field, and MUST only be set for packet that contains Outer-TLVs.  It can be used to calculate the start of the Outer-TLVs.

Version

```
 0 1 2
+-+-+-+
|R|0|1|
+-+-+-+
```

   R = Reserved (must be zero)

Fragmented Message Length

   The Fragmented Message Length field is four octets, and is present only if the L bit is set.  This field provides the total length of the data after the Fragmented Message Length field in the TEAM message or set of messages that is being fragmented.

TLS Message Length

   The TLS Message Length field is four octets, and is present only
   if the T bit is set.  This field provides the total length of the
   TLS Data in the TEAM message.  Data after this length of TLS data
   are the Outer TLVs.

TLS Data

   The TLS data consists of the encapsulated packet in TLS record
   format.

Outer TLVs

   The Outer TLVs consist of the optional data used to help
   establishing the TLS tunnel in TLV format.  The start of the
   Outer-TLVs can be derived from the EAP Length field and TLS Length
   field.

6.  Type-Length-Value Tuples

   The TLVs used within TEAM are standard Type-Length-Value (TLV)
   objects.  The TLV objects could be used to carry arbitrary parameters
   between EAP peer and EAP server.  Possible uses for TLV objects
   include: language and character set for Notification messages and
   cryptographic binding.

   The EAP peer may not necessarily implement all the TLVs supported by
   the EAP server; and hence to allow for interoperability, TLVs allow
   an EAP server to discover if a TLV is supported by the EAP peer,
   using the NAK TLV.  The TEAM packet does not have to contain any
   TLVs, nor need it contain any mandatory TLVs.

   The mandatory bit in a TLV indicates whether support of the TLV is
   required.  If the peer or server does not support the TLV, it MUST
   send a NAK TLV in response, and all the other TLVs in the message
   MUST be ignored.  If an EAP peer or server finds an unsupported TLV
   which is marked as optional, it can ignore the unsupported TLV.  It
   MUST NOT send an NAK TLV.

   Note that a peer or server may support a TLV with the mandatory bit
   set, but may not understand the contents.  The appropriate response
   to a supported TLV with content that is not understood is defined by
   the TLV specification.

   Outer-TLVs SHOULD NOT be included in messages after the first two
   Outer-TLV messages sent by the peer and EAP server respectively.  A
   single Outer-TLV message may be fragmented in multiple TEAM packets.

All Outer-TLVs MUST NOT have the mandatory bit set.  If an Outer-TLV
has the mandatory bit set, then the packet MUST be ignored.

TEAM implementations MUST support TLVs, as well as processing of
mandatory/optional settings on the TLV.

6.1.  TLV Format

TLVs are defined as described below.  The fields are transmitted from
left to right.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|M|R|              TLV Type         |             Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            Value...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
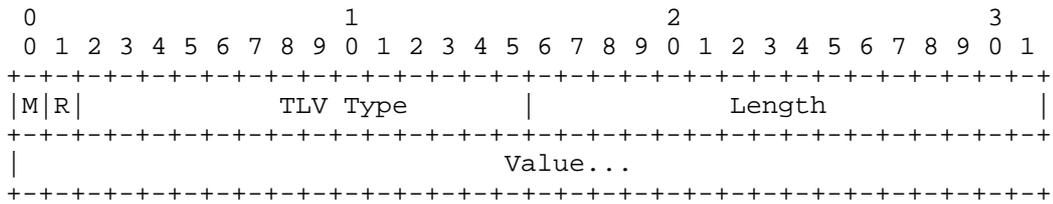
M

    0 -  Optional TLV
    1 -  Mandatory TLV

R

    Reserved, set to zero (0)

TLV Type

    A 14-bit field, denoting the TLV type.  Allocated types include:

    1  -  Result
    2  -  NAK
    3  -  Error-Code
    4  -  Connection-Binding
    5  -  Vendor-Specific
    6  -  URI
    7  -  EAP-Payload
    8  -  Intermediate-Result
    9  -  Crypto-Binding
    10 -  Calling-Station-Id
    11 -  Called-Station-Id
    12 -  NAS-Port-Type

```
       13 -  Server-Identifier
       14 -  Identity-Type
       15 -  Server-Trusted-Root
       16 -  Request-Action
       17 -  PKCS#7
```

   Length

      The length of the Value field in octets

   Value

      The value of the TLV


6.2.  Result TLV

   The Result TLV provides support for acknowledged success and failure
   messages within TEAM.  TEAM implementations MUST support this TLV,
   which cannot be responded to with a NAK TLV.  If the Status field
   does not contain one of the known values, then the peer or EAP server
   MUST drop the connection.  The Result TLV is defined as follows:


```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |M|R|       TLV Type          |              Length              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          Status            |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

   M
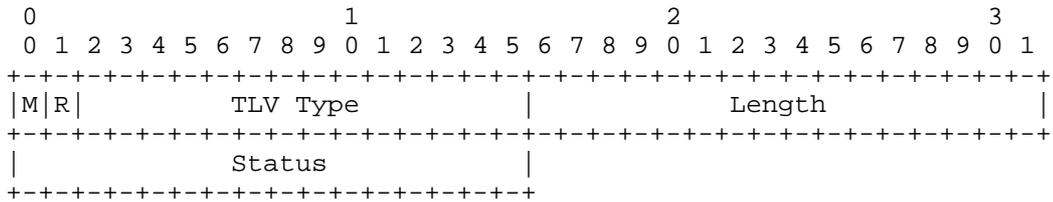
      1 (Mandatory)

   R

      0

   TLV Type

      1 for Result

Length

   2

Status

   The Status field is two octets.  Values include:

   1 -  Success
   2 -  Failure


6.3.  NAK TLV

   The NAK TLV allows a peer to detect TLVs that are not supported by
   the other peer.  A TLV packet can contain 0 or more NAK TLVs.  TEAM
   implementations MUST support the NAK TLV, which cannot be responded
   to with a NAK TLV.  The NAK TLV is defined as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|M|R|        TLV Type          |               Length           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Vendor-Id                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            NAK-Type           |          TLVs....             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

M
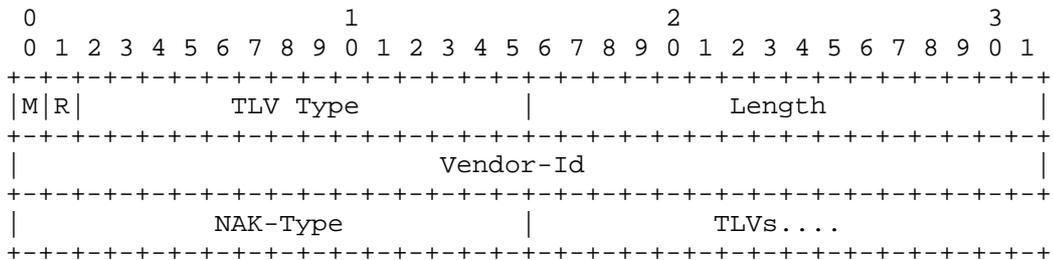
   1 (Mandatory)

R

   0

TLV Type

   2 for NAK

Length

   >= 6

Vendor-ID

   The Vendor-Id field is four octets, and contains the Vendor-Id of
   the TLV that was not supported.  The high-order octet is 0 and the
   low-order 3 octets are the SMI Network Management Private
   Enterprise Code of the Vendor in network byte order.  The Vendor-
   Id field MUST be zero for TLVs that are not Vendor-Specific TLVs.
   For Vendor-Specific TLVs, the Vendor-ID MUST be set to the SMI
   code.

NAK-Type

   The NAK-Type field is two octets.  The field contains the Type of
   the TLV that was not supported.  A TLV of this Type MUST have been
   included in the previous packet.

TLVs

   This field contains a list of TLVs, each of which MUST NOT have
   the mandatory bit set.  These optional TLVs can be used in the
   future to communicate why the offending TLV was determined to be
   unsupported.


6.4.  Error-Code TLV

   The Error-Code TLV allows a TEAM peer or server to indicate errors to
   the other party.  A TLV packet can contain 0 or more Error TLVs.
   Error-Code TLVs MUST be marked as Mandatory.  TEAM implementations
   MUST support the Error-Code TLV, which cannot be responded to with a
   NAK TLV.  The Error-Code TLV is defined as follows:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |M|R|         TLV Type          |              Length           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                          Error-Code                           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

M
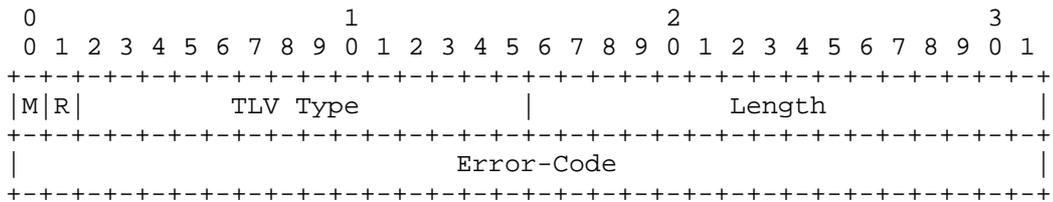
   1 (Mandatory)

   R

      0

   TLV Type

      3 for Error-Code

   Length

      4

   Error-Code

      The Error-Code field is four octets.  Error Codes 1-999 represent
      successful outcomes (informative messages), 1000-1999 represent
      warnings, and codes 2000-2999 represent fatal errors.  If an
      Error- Code TLV with a fatal error has been sent, then the
      conversation must be terminated.

      Currently defined values for Error-Code include:

      2001 -  Tunnel_Compromise_Error
      2002 -  Unexpected_TLVs_Exchanged


6.5.  Crypto-Binding TLV

   The Crypto-Binding TLV is used prove that both peers participated in
   the sequence of authentications (specifically the TLS session and
   inner EAP methods that generate keys).

   Both the Binding Request (B1) and Binding Response (B2) use the same
   packet format.  However the Sub-Type indicates whether it is B1 or
   B2.

   The Crypto-Binding TLV MUST be used to perform Cryptographic Binding
   after each successful EAP method in a sequence of EAP methods is
   complete in TEAM Phase 2.  The Crypto-Binding TLV can also be used
   during Protected Termination.

   The Crypto-Binding TLV must have the version number received during
   the TEAM version negotiation.  The receiver of the Crypto-Binding TLV
   must verify that the version in the Crypto-Binding TLV matches the
   version it sent during the TEAM version negotiation.  If this check
   fails then the TLV is invalid.

   The receiver of the Crypto-Binding TLV must verify that the subtype

is not set to any value other than the ones allowed.  If this check
fails then the TLV is invalid.

This message format is used for the Binding Request (B1) and also the
Binding Response.  This uses TLV type CRYPTO_BINDING_TLV.  TEAM
implementations MUST support this TLV and this TLV cannot be
responded to with a NAK TLV.  The Crypto-Binding TLV is defined as
follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|M|R|         TLV Type          |             Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Reserved   |    Version    |  Received Ver. |   Sub-Type   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                              |
~                            Nonce                             ~
|                                                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                              |
~                        Compound MAC                          ~
|                                                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

M
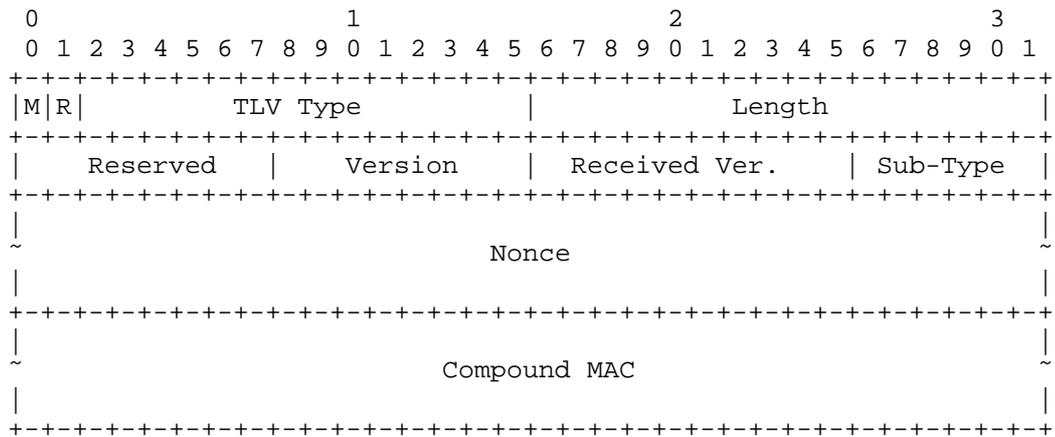
   1 (Mandatory)

R

   0

TLV Type

   9 for Crypto-Binding

Length

   56

Reserved

   Reserved, set to zero (0)

Version

   The Version field is a single octet, which is set to the version
   of Crypto-Binding TLV.  For the Crypto-Binding TLV defined in this
   specification, it is set to one (1).

Sub-Type

   The Sub-Type field is two octets.  Possible values include:

   0 -  Binding Request
   1 -  Binding Response

Nonce

   The Nonce field is 32 octets.  It contains a 256 bit nonce that is
   temporally unique, used for compound MAC key derivation at each
   end.  This is the S_NONCE for the B1 message and a C_NONCE for the
   B2 message.

Compound MAC

   The Compound MAC field is 20 octets.  This can be the Server MAC
   (B1_MAC) or the Client MAC (B2_MAC).  It is computed using the
   HMAC-SHA1-160 keyed MAC that provides 160 bits of output using the
   CMK key.  The MAC is computed over the buffer created after
   concatenating these fields in the following order:

   1.  The entire Crypto-Binding TLV attribute with the MAC field
       zeroed out
   2.  The EAP Type sent by the other party in the first TEAM message
   3.  All the Outer-TLVs from the first TEAM message sent by EAP
       server to peer.  If a single TEAM message is fragmented into
       multiple TEAM packets; then the Outer-TLVs in all the
       fragments of that message MUST be included.
   4.  All the Outer-TLVs from the first TEAM message sent by the
       peer to the EAP server.  If a single TEAM message is
       fragmented into multiple TEAM packets, then the Outer-TLVs in
       all the fragments of that message MUST be included.

6.6.  Connection-Binding TLV

   The Connection-Binding TLV allows for connection specific information
   to be sent by the peer to the AAA server.  This TLV should be logged
   by the EAP or AAA server.  The AAA or EAP server should not deny
   access if there is a mismatch between the value sent through the AAA
   protocol and this TLV.

The format of this TLV is defined for the layer that defines the
parameters.  The format of the value sent by the peer to the EAP
server may be different from the format of the corresponding value
sent through the AAA protocol.  For example, the connection binding
TLV may contain the 802.11 MAC Address or SSID [IEEE.802-11.2007].

TEAM implementations MAY support this TLV and this TLV MUST NOT be
responded to with a NAK TLV.  The Connection-Binding TLV is defined
as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|M|R|         TLV Type          |            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           TLVs...                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

M
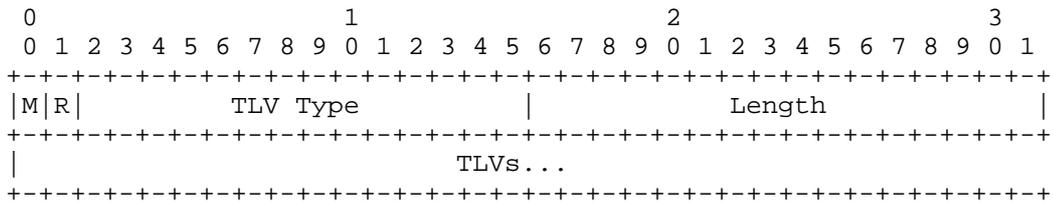
   0 (Optional)

R

   0

TLV Type

   4 for Connection-Binding

Length

   >= 0

TLVs

   The field contains a list of TLVs, each in the same format defined
   in Section 6.1, with the Mandatory flag bit cleared (0).  These
   TLVs contain information on the identity of the peer and
   authenticator (layer 2 or IP addresses); the media used to connect
   the peer and authenticator (NAS-Port-Type); and/or the service the
   client is trying to access on the gateway (SSID).  See
   Section 6.19.4 for further information.

6.7.  Vendor-Specific TLV

   The Vendor-Specific TLV is available to allow vendors to support
   their own extended attributes not suitable for general usage.

   A Vendor-Specific-TLV can contain one or more TLVs, referred to as
   Vendor TLVs.  The TLV-type of the Vendor-TLV will be defined by the
   vendor.  All the Vendor TLVs inside a single Vendor- Specific TLV
   belong to the same vendor.

   TEAM implementations MUST support the Vendor-Specific TLV, and this
   TLV MUST NOT be responded to with a NAK TLV.  If a TEAM
   implementation does not support one or more of the Vendor TLVs inside
   in the Vendor-Specific TLV it SHOULD respond to the Vendor TLV(s)
   with NAK TLV(s) containing the appropriate Vendor-ID and Vendor-TLV
   type.

   Vendor TLVs may be optional or mandatory.  Vendor TLVs sent in the
   protected success and failure packets MUST be marked as optional.  If
   Vendor TLVs sent in protected success/failure packets are marked as
   Mandatory, then the peer or EAP server MUST drop the connection.

   The Vendor-Specific TLV is defined as follows:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |M|R|        TLV Type         |              Length             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                          Vendor-Id                            |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                       Vendor TLVs....                         |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

   M
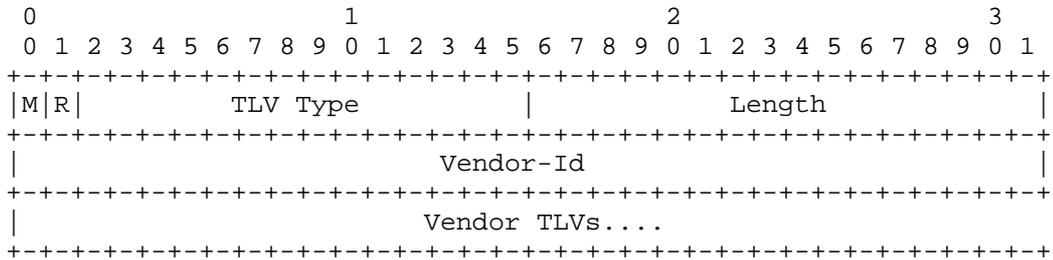
      1 (Mandatory)

   R

      0

   TLV Type

      5 for Vendor-Specific

Length

>= 4

Vendor-ID

The Vendor-Id field is four octets, and contains the Vendor-Id of
the TLV that was not supported.  The high-order octet is 0 and the
low-order 3 octets are the SMI Network Management Private
Enterprise Code of the Vendor in network byte order.  The Vendor-
Id field MUST be zero for TLVs that are not Vendor-Specific TLVs.
For Vendor-Specific TLVs, the Vendor-ID MUST be set to the SMI
code.

Vendor TLVs

This field is of indefinite length.  It contains vendor-
specificTLVs, in a format defined by the vendor.

6.8.  URI TLV

The URI TLV allows a server to send a URI to the client to refer it
to a resource.  The TLV contains a URI in the format specified in RFC
3986 [RFC3986] with UTF-8 encoding.  Interpretation of the value of
the URI is outside the scope of this document.

If a packet contains multiple URI TLVs, then the client SHOULD select
the first TLV it can implement, and ignore the others.  If the client
is unable to implement any of the URI TLVs, then it MAY ignore the
error.  TEAM implementations MAY support this TLV; and this TLV
cannot be responded to with a NAK TLV.  The URI TLV is defined as
follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|M|R|         TLV Type            |             Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              URI                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

M
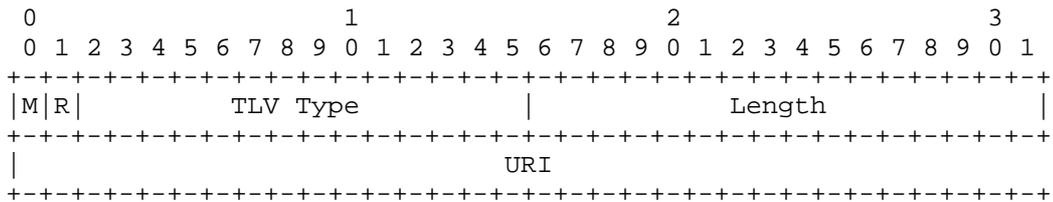
0 (Optional)

R

   0

TLV Type

   6 for URI

Length

   >= 0

URI

   This field is of indefinite length, and conforms to the format
   specified in RFC 3986.

6.9.  EAP-Payload TLV

To allow piggybacking EAP request and response with other TLVs, the
EAP Payload TLV is defined, which includes an encapsulated EAP packet
and 0 or more TLVs.  TEAM implementations MUST support this TLV,
which cannot be responded to with a NAK TLV.  The EAP-Payload TLV is
defined as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|M|R|          TLV Type          |             Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         EAP-Packet...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            TLVs...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

M
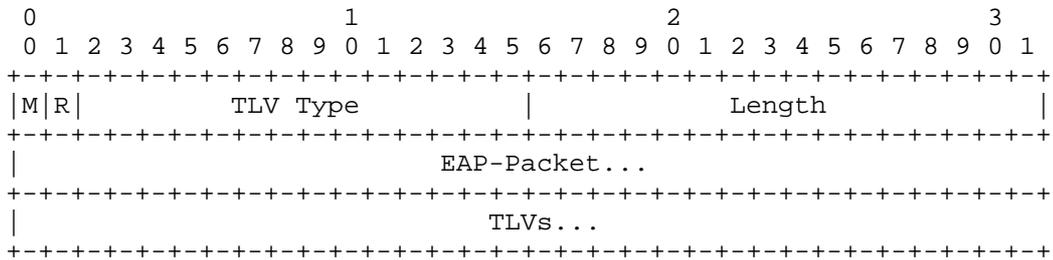
   1 (Mandatory)

R

   0

TLV Type

> 7 for EAP-Payload

Length

> >= 0

EAP-Packet

> This field contains a complete EAP packet, including the EAP
> header (Code, Identifier, Length, Type) fields.  The length of
> this field is determined by the Length field of the encapsulated
> EAP packet.

TLVs

> This (optional) field contains a list of TLVs associated with the
> EAP-Packet field (see Section 6.19.3).  The TLVs utilize the same
> format described Section 6.1, and MUST NOT have the mandatory bit
> set.  The total length of this field is equal to the Length field
> of the EAP- Payload-TLV, minus the Length field in the EAP header
> of the EAP packet field.

## 6.10.  Intermediate-Result TLV

The Intermediate-Result TLV provides support for acknowledged
intermediate Success and Failure messages within EAP.  TEAM
implementations MUST support this TLV, which cannot be responded to
with a NAK TLV.  The Intermediate-Result TLV is defined as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|M|R|        TLV Type         |             Length              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Status          |         TLVs...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

M
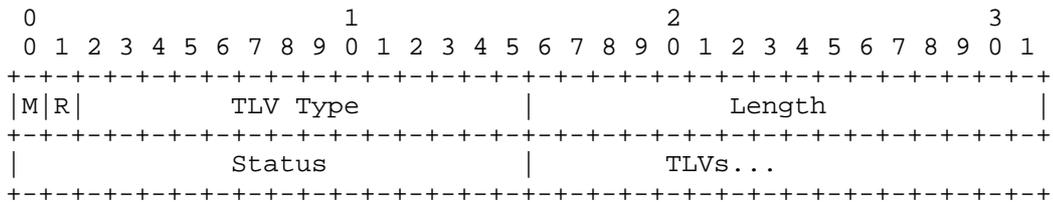
> 1 (Mandatory)

   R

      0

   TLV Type

      8 for Intermediate-Result

   Length

      >= 2

   Status

      The Status field is two octets.  Values include:

         1 - Success
         2 - Failure

   TLVs

      This (optional) field contains a list of TLVs associated with the
      Intermediate-Result TLV.  The TLVs utilize the same format
      described Section 6.1, and MUST NOT have the mandatory bit set.


6.11.  Calling-Station-Id TLV

   This TLV allows a peer to send information to EAP server about the
   call originator.  This TLV MAY be included in the Connection-Binding-
   TLV.

   For dial-up, the Called-Station-ID TLV contains the phone number of
   the peer.  For use with IEEE 802.1X, the MAC address of the peer is
   included [RFC3580].

   For VPN, this attribute is used to send the IPv4 or IPV6 address of
   the interface of the peer used to initiate the VPN in ASCII format.
   Where the Fully Qualified Domain Name (FQDN) of the VPN client is
   known, it SHOULD be appended, separated from the address with a " "
   (space).  Example: "12.20.2.3 vpnserver.example.com".

   As described in Section 7.15 of [RFC3748], this TLV SHOULD be logged
   by the EAP or AAA server, and MAY be used for comparison with
   information gathered by other means.

   However, since the format of this TLV may not match the format of the
   information gathered by other means, if an EAP server or AAA server

supports the capability to deny access based on a mismatch, spurious
authentication failures may occur.  As a result, implementations
SHOULD allow the administrator to disable this check.

TEAM implementations MAY support this TLV and this TLV MUST NOT be
responded to with a NAK TLV.  The Calling-Station-ID TLV is defined
as follows:

```
  0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |M|R|        TLV Type         |              Length             |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                          String...
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

M
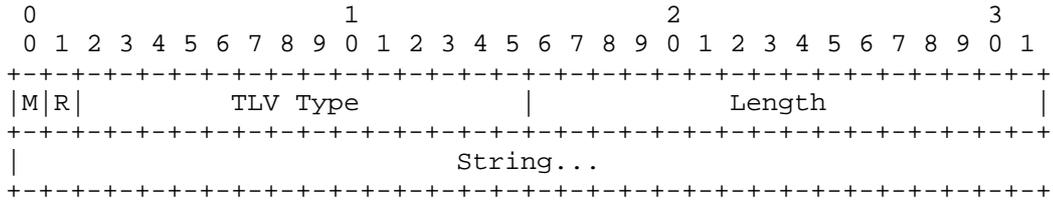
   0 (Optional)

R

   0

TLV Type

   10 for Calling-Station-Id

Length

   >= 0

String

   The field should be the same as the value of the
   Calling-Station-ID attribute in [RFC2865].


6.12.  Called-Station-Id TLV

   This TLV allows a peer to send information to EAP server about the
   NAS it called.  This TLV MAY be included in the Connection-Binding
   TLV.

   For dial-up, the Calling-Station-ID TLV contains the phone number
   called by the peer.  For use with IEEE 802.1X, the MAC address of the
   NAS is included, as specified in [RFC3580].

For VPN, this attribute is used to send the IPv4 or IPv6 address of
VPN server in ASCII format.  Where the Fully Qualified Domain Name
(FQDN) of the VPN server is known, it SHOULD be appended, separated
from the address with a " " (space).  Example: "12.20.2.3
vpnserver.example.com".

This TLV SHOULD be logged by the EAP or AAA server, and MAY be used
for comparison with information gathered by other means.  However,
since the format of this TLV may not match the format of the
information gathered by other means, if an EAP server or AAA server
supports the capability to deny access based on a mismatch, spurious
authentication failures may occur.  As a result, implementations
SHOULD allow the administrator to disable this check.

TEAM implementations MAY support this TLV, and this TLV MUST NOT be
responded to with a NAK TLV.  The Called-Station-ID TLV is defined as
follows:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |M|R|         TLV Type          |             Length            |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                         String...
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

M
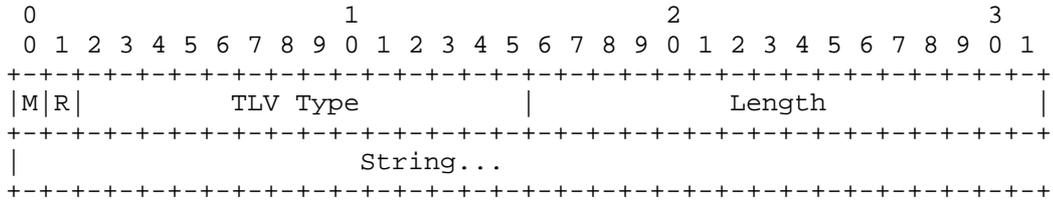
   0 (Optional)

R

   0

TLV Type

   11 for Called-Station-Id

Length

   >= 0

String

   The field should be the same as the value of the Called-Station-ID
   attribute in [RFC2865].

6.13.  NAS-Port-Type TLV

   This TLV allows a peer to send information to EAP server about the
   type of physical connection used by the peer to connect to NAS.  This
   TLV MAY be included in the Connection-Binding-TLV.

   The value of this field is the same as the value of NAS-Port-Type
   attribute in [RFC2865].

   This TLV SHOULD be logged by the EAP or AAA server and MAY be used
   for comparison with information gathered by other means.  However,
   since the format of this TLV may not match the format of the
   information gathered by other means, if an EAP server or AAA server
   supports the capability to deny access based on a mismatch, spurious
   authentication failures may occur.  As a result, implementations
   SHOULD allow the administrator to disable this check.

   TEAM implementations MAY support this TLV; and this TLV MUST NOT be
   responded to with a NAK TLV.  The NAS-Port-Type TLV is defined as
   follows:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |M|R|         TLV Type          |             Length            |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                             Value                             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

   M
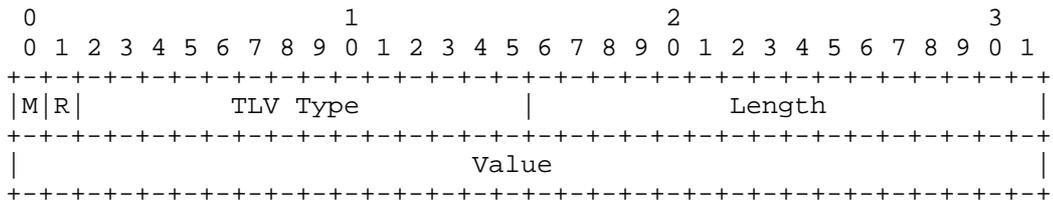
      0 (Optional)

   R

      0

   TLV Type

      12 for NAS-Port-Type

   Length

      4

   String

      The String field is four octets.  Values are the same as for the
      NAS-Port-Type attribute defined in [RFC2865].


6.14.  Server-Identifier TLV

   This TLV allows a EAP server to send a hint to the EAP peer to help
   the EAP peer select the appropriate sessionID for session resumption.
   The field is a string sent by the EAP server, and the field should be
   treated as a opaque string by the peer.  During a full-tls-handshake,
   the EAP peer MAY keep track of this field and the corresponding
   sessionID, and use it as a hint to select the appropriate sessionID
   during session resumption.

   TEAM implementations MAY support this TLV and this TLV MUST NOT be
   responded to with a NAK TLV.  The Server-Identifier TLV is defined as
   follows:


       0                   1                   2                   3
       0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |M|R|         TLV Type          |            Length             |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                           String...
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

   M
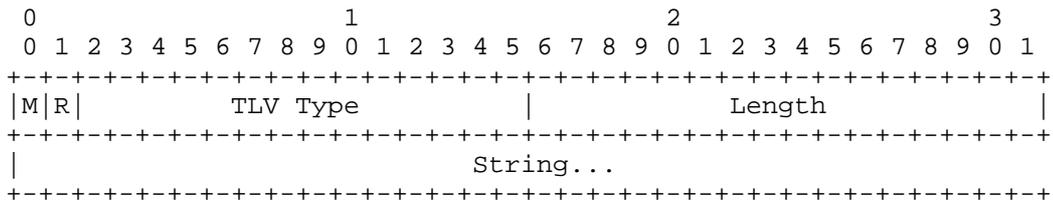
      0 (Optional)

   R

      0

   TLV Type

      13 for Server-Identifier

   Length

      >= 0

      String

         Contains an identifier sent by the EAP server.


6.15.  Identity-Type TLV

      The Identity-Type TLV allows an EAP server to send a hint to help the
      EAP-peer select the right type of identity; for example; user or
      machine.

      TEAM implementations MAY support this TLV, which cannot be responded
      to with a NAK TLV.

      If the Identity Type field does not contain one of the known values
      or if the EAP peer does not have an identity corresponding to the
      identity-type, then the peer MUST ignore the value.  The Identity-
      Type TLV is defined as follows:


       0                   1                   2                   3
       0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |M|R|          TLV Type          |              Length           |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |            Identity Type       |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

   M
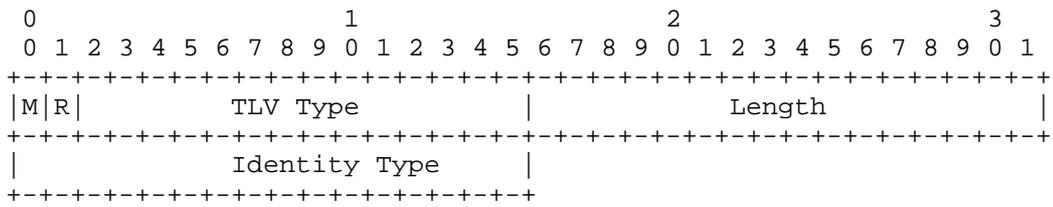
      0 (Optional)

   R

      0

   TLV Type

      14 for Identity-Type

   Length

      2

Identity Type

> The Identity Type field is two octets.  Values include:
>     1 - Human
>     2 - Machine

## 6.16.  Server-Trusted-Root TLV

   The Server-Trusted-Root TLV allows the peer to send a request to the
   EAP server for a trusted root in PKCS #7 format.

   The Server-Trusted-Root TLV is always marked as optional, and cannot
   be responded to with a NAK TLV.  TEAM server implementations that
   claim to support provisioning MUST support Server-Trusted-Root TLV,
   PKCS#7 TLV, and the PKCS#7-Server-Certificate-Root credential format
   defined in this TLV.  TEAM peer implementations may not support this
   TLV.

   The Server-Trusted-Root TLV can only be sent as an inner TLV (inside
   the TEAM Phase 2 conversation), in both server unauthenticated tunnel
   provisioning mode, and the regular authentication process.

   The peer MUST NOT request, or accept the trusted root sent inside the
   Server-Root credential TLV by the EAP server until it has completed
   authentication of the EAP server, and validated the Crypto-Binding
   TLV.  The peer may receive a trusted root, but is not required to use
   the trusted root received from the EAP server.

   If the EAP server sets credential-format to PKCS#7-Server-
   Certificate-Root, then the Server-Trusted-Root TLV MUST contain the
   root of the certificate chain of the certificate issued to the EAP
   server packages in a PKCS#7 TLV.  If the Server certificate is a
   self-signed certificate, then the root is the self-signed
   certificate.  In this case, the EAP server does not have to sign the
   certificate inside the PCKS#7 TLV since it does not necessarily have
   access to the private key for it.

   If the Server-Trusted-Root TLV credential format does not contain one
   of the known values, then the EAP server MUST ignore the value.

   The Server-Trusted-Root TLV is defined as follows:

```
       0                   1                   2                   3
       0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |M|R|         TLV Type          |              Length           |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |         Credential Type       |     TLVs...
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
```

   M
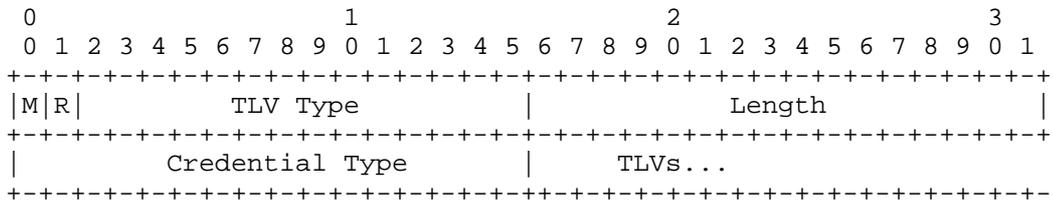
      0 (Optional)

   R

      0

   TLV Type

      15 for Server-Trusted-Root

   Length

      >= 2

   Credential Type

      The Credential Type field is two octets.  Values include:
         1 - PKCS#7-Server-Certificate-Root

   TLVs

      This (optional) field contains a list of TLVs associated with the
      Server-Trusted-Root TLV.  The TLVs utilize the same format
      described Section 6.1 and MUST NOT have the mandatory bit set.
      See Section 6.19.5 for further information.


6.17.  PKCS#7 TLV

   This TLV contains a certificate or certificate chain requested by the
   peer in PKCS#7 format [RFC2315].

   The PKCS#7 TLV is always marked as optional, and cannot be responded
   to with a NAK TLV.  TEAM server implementations that claim to support
   provisioning MUST support this TLV.  TEAM peer implementations may
   not support this TLV.

   If the PKCS#7 TLV contains a certificate or certificate chain that is

not acceptable to the peer, then peer MUST ignore the value.

The PKCS#7 TLV is defined as follows:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |M|R|          TLV Type           |             Length          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |              PKCS#7 data...
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

M
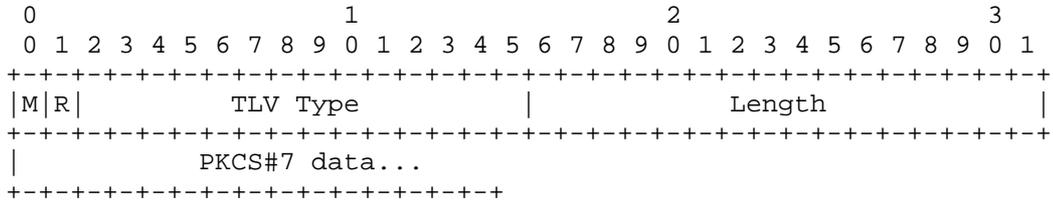
   0 (Optional)

R

   0

TLV Type

   17 for PKCS#7

Length

   >= 0

PKCS#7 Data

   This field contains a certificate or certificate chain in PKCS#7
   format.


6.18.  Request-Action-TLV

   The Request-Action TLV MAY be sent by the peer along with
   acknowledged failure.  It allows the peer to request the EAP server
   to negotiate EAP methods or process TLVs specified in the failure
   packet.  The server MAY ignore this TLV.

   TEAM implementations MUST support this TLV, which cannot be responded
   to with a NAK TLV.

   The Request-Action TLV is defined as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|M|R|         TLV Type         |              Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              Action          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

M
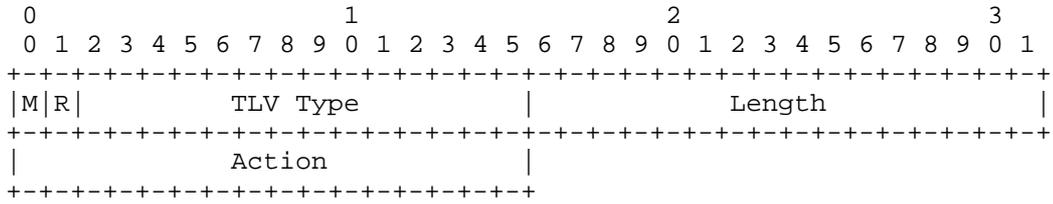
   1 (Mandatory)

R

   0

TLV Type

   16 for Request-Action

Length

   2

Action

   The Action field is two octets.  Values include:

      0 - Reserved
      1 - Process-TLV
      2 - Negotiate-EAP


6.19.  TLV Rules

   To save round trips, multiple TLVs can be sent in the single TEAM
   packet.  However, the encapsulation of multiple EAP Payload TLVs
   within a single TEAM packet is not supported in this version and MUST
   NOT be sent.  If the peer or EAP server receives multiple EAP Payload
   TLVs, then it MUST drop the connection.

The following table defines the meaning of the table entries in the
sections below:

0    This TLV MUST NOT be present in the packet
0+   Zero or more instances of this TLV MAY be present in packet
0-1  Zero or one instances of this TLV MAY be present in packet
1    Exactly one instance of this TLV MUST be present in packet

6.19.1.  Outer TLVs

The following table provides a guide to which TLVs may be included in
the TEAM packet outside the TLS channel, which kind of packets, and
in what quantity:

| Request | Response | Success | Failure | TLV in unencrypted-TLVs field |
|---------|----------|---------|---------|-------------------------------|
| 0-1     | 0        | 0       | 0       | Server-Identifier TLV         |
| 0+      | 0+       | 0       | 0       | Vendor-Specific TLV           |

Outer-TLVs MUST be marked as optional.  Vendor-TLVs inside a Vendor-
Specific TLV MUST be marked as optional when included in Outer TLVs.
Outer-TLVs MUST NOT be included in messages after the first two TEAM
messages sent by peer and EAP server respectively, i.e., the first
EAP server to peer message and first peer to EAP server message.  If
a message is fragmented, the whole set of fragments is counted as one
message.  If Outer-TLVs are included in messages after the first two
TEAM messages, they MUST be ignored.

6.19.2.  Inner TLVs

   The following table provides a guide to which Inner TLVs may be
   encapsulated in TLS in TEAM Phase 2, in which kind of packets, and in
   what quantity:

```
+---------+----------+---------+---------+--------------------+
| Request | Response | Success | Failure | Inner TLV          |
+---------+----------+---------+---------+--------------------+
| 0-1     | 0-1      | 0-1     | 0-1     | Intermediate-Result |
| 0-1     | 0-1      | 0       | 0       | EAP-Payload        |
| 0-1     | 0-1      | 1       | 1       | Result             |
| 0-1     | 0-1      | 1       | 1       | Crypto-Binding     |
| 0+      | 0+       | 0+      | 0+      | Error              |
| 0+      | 0+       | 0       | 0       | NAK                |
| 0-1     | 0-1      | 0-1     | 0-1     | Connection-Binding |
| 0+      | 0+       | 0+      | 0+      | Vendor-Specific    |
| 0+      | 0        | 0+      | 0-1     | URI                |
| 0+      | 0        | 0       | 0       | Identity-Type      |
| 0+      | 0+       | 0+      | 0+      | Server-Trusted-Root |
| 0       | 0-1      | 0       | 0-1     | Request-Action     |
+---------+----------+---------+---------+--------------------+
```

   Vendor TLVs (included in Vendor-Specific TLVs) sent in the protected
   success and failure packets MUST be marked as optional.  If Vendor
   TLVs sent in protected success/failure packets are marked as
   Mandatory, then the peer or EAP server MUST drop the connection.

```
+----------+------------------------------------------------------+
| Packet   | Description                                          |
| Type     |                                                      |
+----------+------------------------------------------------------+
| Request  | TLS packet sent by the EAP server to the peer        |
| Response | TLS packet sent by the peer to the EAP server        |
| Success  | TLS packet sent by the peer or EAP server as a       |
|          | protected success indication                         |
| Failure  | TLS packet sent by the peer or EAP server as a       |
|          | protected failure indication                         |
+----------+------------------------------------------------------+
```

6.19.3.  EAP-Payload TLV

   The EAP-Payload TLV can contain other TLVs.  The table below defines
   which TLVs can be contained inside the EAP-Payload TLV and how many
   such TLVs can be included.

```
   +---------+----------+-----------------+
   | Request | Response | TLV             |
   +---------+----------+-----------------+
   | 0+      | 0+       | Vendor-Specific |
   | 0+      | 0+       | Identity-Type   |
   +---------+----------+-----------------+
```

   Vendor TLVs encapsulated in a Vendor-Specific TLV MUST be marked as
   optional when included in an EAP-Payload TLV.

6.19.4.  Connection-Binding TLV

   The Connection-Binding TLV can contain other TLVs.  The table below
   defines which TLVs can be contained inside the Connection-Binding TLV
   and how many such TLVs can be included.

```
   +---------+----------+-------------------+
   | Request | Response | TLV               |
   +---------+----------+-------------------+
   | 0-1     | 0        | Calling-Station-ID |
   | 0-1     | 0        | Called-Station-ID  |
   | 0-1     | 0        | NAS-Port-Type      |
   | 0+      | 0+       | Vendor-Specific   |
   +---------+----------+-------------------+
```

   Vendor TLVs encapsulated in a Vendor-Specific TLV MUST be marked as
   optional when included in a Connection-Binding TLV.

6.19.5.  Server-Trusted-Root TLV

   The Server-Trusted-Root TLV can contain other TLVs.  The table below
   defines which TLVs can be contained inside the Server-Trusted-Root
   TLV and how many such TLVs can be included.

```
   +---------+----------+--------+
   | Request | Response | TLV    |
   +---------+----------+--------+
   | 0-1     | 0        | PKCS#7 |
   +---------+----------+--------+
```

7.  Security Considerations

7.1.  Authentication and Integrity Protection

   TEAM provides a server authenticated, encrypted and integrity
   protected tunnel.  All data within the tunnel has these properties.
   Data outside the tunnel such as EAP Success and Failure, Outer-TLVs,
   authentication methods negotiated outside of TEAM and the TEAM
   headers themselves (including the EAP-Type in the header) are not
   protected by this tunnel.

   In addition, the Crypto-Binding TLV can reveal a man-in-the-middle
   attack described in Section 7.8, below.  Hence, the server should not
   reveal any sensitive data to the client until after the Crypto-
   Binding TLV has been properly verified.

   In order to detect the modification of Outer TLVs, the first two
   Outer TLV messages sent by both peer and EAP server are included in
   the calculation of the Crypto-Binding TLV.  Outer-TLVs SHOULD NOT be
   included in other TEAM packets since there is no mechanism to detect
   modification.

   In order to detect modification of EAP-Type sent in the clear (EAP-
   Type should be set to TEAM), the EAP-Type sent in the first two
   messages by both peer and EAP server is included in the calculation
   of Crypto-Binding TLV.  The EAP-Type in the clear could be modified
   in other TEAM packets and will likely result in failure, hence it is
   not included in the Crypto-Binding calculation.

7.2.  Method Negotiation

   If the peer does not support TEAM, or does not wish to utilize TEAM
   authentication, it MUST respond to the initial EAP- Request/
   TEAM-Start with a NAK, suggesting an alternate authentication method.
   Since the NAK is sent in cleartext with no integrity protection or
   authentication, it is subject to spoofing.  Inauthentic NAK packets
   can be used to trick the peer and authenticator into "negotiating
   down" to a weaker form of authentication, such as EAP- MD5 (which
   only provides one way authentication and does not derive a key).

   Since a subsequent protected EAP conversation can take place within
   the TLS session, selection of TEAM as an authentication method does
   not limit the potential secondary authentication methods.  As a
   result, the only legitimate reason for a peer to NAK TEAM as an
   authentication method is that it does not support it.  Where the
   additional security of TEAM is required, server implementations
   SHOULD respond to a NAK with an EAP-Failure, terminating the
   authentication conversation.

Since method negotiation outside of TEAM is not protected, if the
peer is configured to allow TEAM and other EAP methods at the same
time, the negotiation is subject to downgrade attacks.  Since method
negotiation outside of TEAM is not protected, if the peer is
configured to allow TEAM and previous TEAM versions at the same time,
the negotiation is subject to negotiation downgrade attacks.
However, peers configured to allow TEAM and later TEAM versions may
not be subject to downgrade negotiation attack since the highest
version supported by both peers is checked within the protected
tunnel.

If peer implementations select incorrect methods or credentials with
EAP servers, then attacks are possible on the credentials.  Hence, a
TEAM peer implementation should preferably be configured with a set
of credentials and methods that may be used with a specific TEAM
server.  The peer implementation may be configured to use different
methods and/or credentials based on the TEAM server.

7.3.  TLS Session Cache Handling

In cases where a TLS session has been successfully resumed, in some
circumstances, it is possible for the EAP server to skip TEAM Phase
2, and successfully conclude the conversation with a protected
termination.

TEAM "fast reconnect" is desirable in applications such as wireless
roaming, since it minimizes interruptions in connectivity.  It is
also desirable when the "inner" EAP mechanism used is such that it
requires user interaction.  The user should not be required to re-
authenticate herself, using biometrics, token cards or similar, every
time the radio connectivity is handed over between access points in
wireless environments.

However, there are issues that need to be understood in order to
avoid introducing security vulnerabilities.

Since Phase 1 of TEAM may not provide client authentication,
establishment of a TLS session (and an entry in the TLS session
cache) does not by itself provide an indication of the peer's
authenticity.

Some TEAM implementations may not be capable of removing TLS session
cache entries established in TEAM Phase 1 after an unsuccessful Phase
2 authentication.  In such implementations, the existence of a TLS
session cache entry provides no indication that the peer has
previously been authenticated.  As a result, implementations that do
not remove TLS session cache entries after a TEAM Phase 2
authentication or failed protected termination MUST use other means

than successful TLS resumption as the indicator of whether the client
is authenticated or not.  The implementation MUST determine that the
client is authenticated only after the completion of protected
termination.  Failing to do this would enable a peer to gain access
by completing TEAM Phase 1, tearing down the connection, re-
connecting and resuming TEAM Phase 2, thereby proving herself
authenticated.  Thus, TLS resumption MUST only be enabled if the
implementation supports TLS session cache removal.  If an EAP server
implementing TEAM removes TLS session cache entries of peers failing
TEAM Phase 2 authentication, then it MAY skip the TEAM Phase 2
conversation entirely after a successful session resumption,
successfully terminating the TEAM conversation as described in
Section 4.4.2.

7.4.  Certificate Revocation

   Since the EAP server usually has network connectivity during the EAP
   conversation, the server is capable of following a certificate chain
   or verifying whether the peer's certificate has been revoked.  In
   contrast, the peer may or may not have network connectivity, and thus
   while it can validate the EAP server's certificate based on a pre-
   configured set of CAs, it may not be able to follow a certificate
   chain or verify whether the EAP server's certificate has been
   revoked.

   In the case where the peer is initiating a voluntary Layer 2 channel
   using PPTP [RFC2637] or L2TP [RFC3931], the peer will typically
   already have network connectivity established at the time of channel
   initiation.  As a result, during the EAP conversation it is capable
   of checking for certificate revocation.

   As part of the TLS negotiation, the server presents a certificate to
   the peer.  The peer SHOULD verify the validity of the EAP server
   certificate, and SHOULD also examine the EAP server name presented in
   the certificate, in order to determine whether the EAP server can be
   trusted.  Please note that in the case where the EAP authentication
   is remote, the EAP server will not reside on the same machine as the
   authenticator, and therefore the name in the EAP server's certificate
   cannot be expected to match that of the intended destination.  In
   this case, a more appropriate test might be whether the EAP server's
   certificate is signed by a CA controlling the intended destination
   and whether the EAP server exists within a target sub-domain.

   In the case where the peer is attempting to obtain network access, it
   will not have network connectivity.  The TLS Extensions [RFC5246]
   support piggybacking of an Online Certificate Status Protocol
   [RFC2560] or a Server-based Certificate Validation Protocol [RFC5055]
   response within TLS, therefore can be utilized by the peer in order

to verify the validity of server certificate.  However, since not all
TLS implementations implement the TLS extensions, it may be necessary
for the peer to wait to check for certificate revocation until after
network access has been obtained.  In this case, the peer SHOULD
conduct the certificate status check immediately upon going online
and SHOULD NOT send data until it has received a positive response to
the status request.  If the server certificate is found to be invalid
as per client policy, then the peer SHOULD disconnect.

If the client has a policy to require checking certificate revocation
and it cannot obtain revocation information then it may need to
disallow the use of all or some of the inner methods since some
methods may reveal some sensitive information.

7.5.  Separation of EAP Server and Authenticator

As a result of a complete TEAM conversation, the EAP endpoints will
mutually authenticate, and derive a session key for subsequent use in
link layer security.  Since the peer and EAP client reside on the
same machine, it is necessary for the EAP client module to pass the
session key to the link layer encryption module.

The situation may be more complex on the Authenticator, which may or
may not reside on the same machine as the EAP server.  In the case
where the EAP server and the Authenticator reside on different
machines, there are several implications for security.  Firstly, the
mutual authentication defined in TEAM will occur between the peer and
the EAP server, not between the peer and the authenticator.  This
means that as a result of the TEAM conversation, it is not possible
for the peer to validate the identity of the NAS or channel server
that it is speaking to.

The second issue is that the session key negotiated between the peer
and EAP server will need to be transmitted to the authenticator.
Therefore a secure mechanism needs to be provided to transmit the
session key from the EAP server to the authenticator or channel
server that needs to use the key.  The specification of this transit
mechanism is outside the scope of this document.

7.6.  Separation of TEAM Phase 1 and 2 Servers

The EAP server involved in TEAM Phase 2 need not necessarily be the
same as the EAP server involved in Phase 1.  For example, a local
authentication server or proxy might serve as the endpoint for the
Phase 1 conversation, establishing the TLS channel.  Subsequently,
once the EAP-Response/Identity has been received within the TLS
channel, it can be decrypted and forwarded in cleartext to the
destination realm EAP server.  The rest of the conversation will

therefore occur between the destination realm EAP server and the
peer, with the local authentication server or proxy acting as an
encrypting/decrypting gateway.  This permits a non-TLS capable EAP
server to participate in the TEAM conversation.

Note however that such an approach introduces security
vulnerabilities.  Since the EAP Response/Identity is sent in the
clear between the proxy and the EAP server, this enables an attacker
to snoop the user's identity.  It also enables a remote environments,
which may be public hot spots or Internet coffee shops, to gain
knowledge of the identity of their users.  Since one of the potential
benefits of TEAM is identity protection, this is undesirable.

If the EAP method negotiated during TEAM Phase 2 does not support
mutual authentication, then if the Phase 2 conversation is proxied to
another destination, the TEAM peer will not have the opportunity to
verify the secondary EAP server's identity.  Only the initial EAP
server's identity will have been verified as part of TLS session
establishment.

Similarly, if the EAP method negotiated during TEAM Phase 2 is
vulnerable to dictionary attack, then an attacker capturing the
cleartext exchange will be able to mount an offline dictionary attack
on the password.

Finally, when a Phase 2 conversation is terminated at a different
location than the Phase 1 conversation, the Phase 2 destination is
unaware that the EAP client has negotiated TEAM.  As a result, it is
unable to enforce policies requiring TEAM.  Since some EAP methods
require TEAM in order to generate keys or lessen security
vulnerabilities, where such methods are in use, such a configuration
may be unacceptable.

In summary, TEAM encrypting/decrypting gateway configurations are
vulnerable to attack and SHOULD NOT be used.  Instead, the entire
TEAM connection SHOULD be proxied to the final destination, and the
subsequently derived master session keys need to be transmitted back.
This provides end-to-end protection of TEAM.  The specification of
this transit mechanism is outside the scope of this document, but
mechanisms similar to those described in [RFC2548] can be used.
These steps protect the client from revealing her identity to the
remote environment.

In order to find the proper TEAM destination, the EAP client SHOULD
place a Network Access Identifier (NAI) [RFC4282] in the Identity
Response.

There may be cases where a natural trust relationship exists between

the (foreign) authentication server and final EAP server, such as on
a campus or between two offices within the same company, where there
is no danger in revealing the identity of the station to the
authentication server.  In these cases, a proxy solution without end
to end protection of TEAM MAY be used.  If RADIUS [RFC2865] is used
to communicate between gateway and EAP server, then the TEAM
encrypting/decrypting gateway SHOULD provide support for IPsec
protection of RADIUS in order to provide confidentiality for the
portion of the conversation between the gateway and the EAP server,
as described in [RFC3579].

7.7.  Identity Verification

   Since the TLS session has not yet been negotiated, the initial
   Identity request/response occurs in the clear without integrity
   protection or authentication.  It is therefore subject to snooping
   and packet modification.

   In configurations where all users are required to authenticate with
   TEAM and the first portion of the TEAM conversation is terminated at
   a local backend authentication server, without routing by proxies,
   the initial cleartext Identity Request/Response exchange is not
   needed in order to determine the required authentication method(s) or
   route the authentication conversation to its destination.  As a
   result, the initial Identity and Request/Response exchange may not be
   present, and a subsequent Identity Request/Response exchange MAY
   occur after the TLS session is established.

   If the initial cleartext Identity Request/Response has been tampered
   with, after the TLS session is established, it is conceivable that
   the EAP Server will discover that it cannot verify the peer's claim
   of identity.  For example, the peer's userID may not be valid or may
   not be within a realm handled by the EAP server.  Rather than
   attempting to proxy the authentication to the server within the
   correct realm, the EAP server SHOULD terminate the conversation.

   The TEAM peer can present the server with multiple identities.  This
   includes the claim of identity within the initial EAP- Response/
   Identity (MyID) packet, which is typically used to route the EAP
   conversation to the appropriate home backend authentication server.
   There may also be subsequent EAP-Response/Identity packets sent by
   the peer once the TLS channel has been established.

   Note that since the TEAM peer may not present a certificate, it is
   not always possible to check the initial EAP-Response/Identity
   against the identity presented in the certificate, as is done in
   [RFC5216].

Moreover, it cannot be assumed that the peer identities presented
within multiple EAP-Response/Identity packets will be the same.  For
example, the initial EAP-Response/Identity might correspond to a
machine identity, while subsequent identities might be those of the
user.  Thus, TEAM implementations SHOULD NOT abort the authentication
just because the identities do not match.  However, since the initial
EAP-Response/Identity will determine the EAP server handling the
authentication, if this or any other identity is inappropriate for
use with the destination EAP server, there is no alternative but to
terminate the TEAM conversation.

The protected identity or identities presented by the peer within
TEAM Phase 2 may not be identical to the cleartext identity presented
in TEAM Phase 1, for legitimate reasons.  In order to shield the
userID from snooping, the cleartext Identity may only provide enough
information to enable routing of the authentication request to the
correct realm.  For example, the peer may initially claim the
identity of "nouser@bigco.com" in order to route the authentication
request to the bigco.com EAP server.  Subsequently, once the TLS
session has been negotiated, in TEAM Phase 2, the peer may claim the
identity of "fred@bigco.com".  Thus, TEAM can provide protection for
the user's identity, though not necessarily the destination realm,
unless the TEAM Phase 1 conversation terminates at the local
authentication server.

As a result, TEAM implementations SHOULD NOT attempt to compare the
Identities claimed with Phases 1 and 2 of the TEAM conversation.
Similarly, if multiple Identities are claimed within TEAM Phase 2,
these SHOULD NOT be compared.  An EAP conversation may involve more
than one EAP authentication method, and the identities claimed for
each of these authentications could be different (e.g. a machine
authentication, followed by a user authentication).

7.8.  Man-in-the-Middle Attack Protection

   TLS protection can address a number of weaknesses in the EAP method;
   as well as EAP protocol weaknesses listed in the abstract and
   introduction sections in this document.

   Hence, the recommended solution is to always deploy authentication
   methods with protection of TEAM.

   If a deployment chooses to allow a EAP method protected by TEAM
   without protection of TEAM or IPsec at the same time, then this opens
   up a possibility of a man-in-the-middle attack.

   A man-in-the-middle can spoof the client to authenticate to it
   instead of the real EAP server; and forward the authentication to the

real server over a protected tunnel.  Since the attacker has access
to the keys derived from the tunnel, it can gain access to the
network.

TEAM prevents this attack by using the keys generated by the inner
EAP method in the crypto-binding exchange described in protected
termination section.  This attack is not prevented if the inner EAP
method does not generate keys or if the keys generated by the inner
EAP method can be compromised.  Hence, in cases where the inner EAP
method does not generate keys, the recommended solution is to always
deploy authentication methods protected by TEAM.

Alternatively, the attack can also be thwarted if the inner EAP
method can signal to the peer that the packets are being sent within
the tunnel.  In most cases this may require modification to the inner
EAP method.  In order to allow for these implementations, TEAM
implementations should inform inner EAP methods that the EAP method
is being protected by a TEAM tunnel.

Since all sequence negotiations and exchanges are protected by TLS
channel, they are immune to snooping and MITM attacks with the use of
Crypto-Binding TLV.  To make sure the same parties are involved
tunnel establishment and previous inner method, before engaging the
next method to sent more sensitive information, both peer and server
MUST use the Crypto-Binding TLV between methods to check the tunnel
integrity.  If the Crypto-Binding TLV failed validation, they SHOULD
stop the sequence and terminate the tunnel connection, to prevent
more sensitive information being sent in subsequent methods.

7.9.  Cleartext Forgeries

As described in [RFC3748], EAP Success and Failure packets are not
authenticated, so that they may be forged by an attacker without fear
of detection.  Forged EAP Failure packets can be used to convince an
EAP peer to disconnect.  Forged EAP Success and Failure packets may
be used to convince a peer to disconnect; or convince a peer to
access the network even before authentication is complete, resulting
in denial of service for the peer.

By supporting encrypted, authenticated and integrity protected
success/failure indications, TEAM provides protection against these
attacks.

Once the peer responds with the first TEAM packet; and the EAP server
receives the first TEAM packet from the peer, both MUST silently
discard all clear text EAP messages unless both the TEAM peer and
server have indicated success or failure or error using a protected
error or protected termination mechanism.  The success/failure

decisions sent by a protected mechanism indicate the final decision
of the EAP authentication conversation.  After success/failure has
been indicated by a protected mechanism, the TEAM client can process
unprotected EAP success and EAP failure message; however MUST ignore
any unprotected EAP success or failure messages where the decision
does not match the decision of the protected mechanism.

After a Fatal alert is received or after protected termination is
complete, the peer or EAP server should accept clear text EAP
messages.  If the TEAM tunnel is nested inside another tunnel, then
the clear text EAP messages should only be accepted after protected
termination of outer tunnels.

RFC 3748 states that an EAP Success or EAP Failure packet terminates
the EAP conversation, so that no response is possible.  Since EAP
Success and EAP Failure packets are not retransmitted, if the final
packet is lost, then authentication will fail.  As a result, where
packet loss is expected to be non-negligible, unacknowledged success/
failure indications lack robustness.

As a result, a EAP server SHOULD send a clear text EAP Success or
Failure packet after the protected success or failure packet or TLS
alert.  The peer MUST NOT require the clear text EAP Success or EAP
Failure if it has received the protected success or failure or TLS
alert.  For more details, refer to Section 4.2 of RFC 3748.

7.10.  TLS Ciphersuites

Anonymous ciphersuites are vulnerable to man-in-the-middle attacks,
and SHOULD NOT be used with TEAM, unless the EAP methods inside TEAM
can address the man-in-the-middle attack or unless the man-in- the-
middle attack can be addressed by mechanisms external to TEAM.

7.11.  Denial of Service Attacks

Denial of service attacks are possible if the attacker can insert or
modify packets in the authentication channel.  The attacker can
modify unprotected fields in the TEAM packet such as the EAP protocol
or TEAM version number.  This can result in a denial of service
attack.  It is also possible for the attacker to modify protected
fields in a packet to cause decode errors resulting in a denial of
service.  In these ways the attacker can prevent access for peers
connecting to the network.

Denial of service attacks with multiplier impacts are more
interesting than the ones above.  It is possible to multiply the
impact by creating a large number of TLS sessions with the EAP
server.

7.12.  Server Unauthenticated Tunnel Provisioning Mode

   This section describes the rationale and security risks behind server
   unauthenticated tunnel provisioning mode.  Server unauthenticated
   tunnel provisioning mode can result in potential security
   vulnerabilities.  Hence, this mode is optional in TEAM
   implementations.

   In order to achieve strong mutual authentication, it is best to use
   an out of band mechanism to pre-provision the device with strong
   symmetric or asymmetric keys.  In addition, if the device is not
   physically secure (mobile or devices at public places), then it is
   important to ensure that the device has secure storage.

   Server unauthenticated tunnel provisioning mode is not recommended
   for use in devices which already support secure provisioning and
   secure credential storage capabilities.

   If the provisioned credential is a shared key or asymmetric key
   issued to the peer, then the credential should only be issued to
   devices that can protect the provisioned credentials using secure
   storage, or use physical security.

   If the credentials are not protected, the attacker can compromise the
   provisioned credentials, and use them to get access to the network.
   Mobile light weight devices are typically not physically secure.
   Another concern is that credentials provisioned to a light weight
   mobile device that does not use secure storage could be transferred
   to a general operating system and used to get access to the network.

   If the provisioned credential is a certificate trusted root of the
   EAP server, this is public information and hence not susceptible to
   the same attacks as a shared key or asymmetric key.

   In server unauthenticated tunnel provisioning mode, an attacker may
   terminate the tunnel instead of the real server.  The attacker can be
   detected after the Crypto-Binding TLV is exchanged and validated.
   However, the EAP packets exchanged inside the tunnel until Crypto-
   Binding TLV is validated are available in unencrypted form to the
   attacker.  It is difficult to completely negate the security risk
   unless the EAP methods inside the tunnel are secure; or unless
   physical wire security is assumed.

   The standard credential request/response capability is designed to be
   independent of the server unauthenticated tunnel provisioning mode,
   and can be used in regular authentication mode to provision other
   credentials to the peer that can be used for authentication to the
   network, or for potentially authentications to other services.

The security risks vary depending on the type of credential
exchanged, the scope of use of the credential, and the implementation
of the device.

These are a few guidelines to reduce the security risk:

1.  Minimize the use of this mode only during initial authentication
    to the network to reduce the risk of attack

2.  The password-based EAP method used in provisioned mode MUST be
    resistant to dictionary attacks

3.  Disable this mode by default and require users to initiate
    provisioning mode explicitly rather than being prompted during
    initiation of regular authentication process

4.  Provide appropriate policy capabilities to allow administrators
    to lockdown the device and prevent regular users from enabling
    the mode

5.  Ensure that the EAP methods used support mutual authentication,
    key derivation and resistance to dictionary attack

6.  Ensure that the keys generated by EAP methods are of sufficient
    strength to prevent compounding binding from being compromised

7.  Minimize the information disclosed to the EAP server


7.13.  Security Claims

   Intended use:            Wireless or Wired networks, and over the
                            Internet, where physical security cannot be
                            assumed

   Authentication mechanism: Uses arbitrary EAP and TLS authentication
                            mechanisms for authentication of the client
                            and server.

   Ciphersuite negotiation: Yes

   Mutual authentication:   Yes (depends on the type of EAP method used
                            within the tunnel and the type of
                            authentication used within TLS)

      Integrity protection:    Yes

      Replay protection:       Yes

      Confidentiality:         Yes

      Key derivation:          Yes

      Key strength:            Variable

      Dict. attack protection: Yes

      Fast reconnect:          Yes

      Cryptographic binding:   Yes

      Acknowledged S/F:        Yes

      Session independence:    Yes

      Fragmentation:           Yes

      State synchronization:   Yes

   The TEAM protocol is unconditionally compliant with the requirements
   for WLAN authentication mechanisms, as specified in [RFC4017].

   TEAM derives keys by combining keys from TLS and the inner EAP
   methods.  It should be noted that the use of TLS ciphersuites with a
   particular key lengths does not guarantee that the key strength of
   the keys will be equivalent to the length.  The key exchange
   mechanisms (e.g., RSA or Diffie-Hellman) used must provide sufficient
   security or they will be the weakest link.  For example, RSA key
   sizes with a modulus of 1024 bits provides less than 128 bits of
   security; this may provide sufficient key strength for some
   applications and not for others.  See BCP 86 [RFC3766] for a detailed
   analysis of the strength requirements on the public keys used to
   exchange symmetric keys.

8.  IANA Considerations

   This memo specifies new values and registries to be created and
   managed by IANA.  The policies used to allocate numbers are described
   in [RFC5226].

8.1.  EAP Type

   This memo requires IANA to allocate a new EAP method type for TEAM.
   The placeholder indicated by <TBD> in section Section 5.2 above shall
   be replaced by the new EAP method type upon assignment by IANA.

8.2.  TLV Types

   IANA is requested to create a registry for TEAM TLV Types.

   TLV Types may assume a value between 0 and 16383 of which 0-20 are
   allocated in this document Section 6.  Additional TLV type codes may
   be allocated following the "Specification Required" policy [RFC5226].

8.3.  TLV Values

   IANA is requested to create a registry for TEAM TLV Values, populated
   initially with entries for the Identity-Type, Credential Type and
   Action fields.

   The Identity-Type field may assume a value between 0 and 65535, of
   which 0-2 are allocated in this document Section 6.15, Additional
   Identity-Type values may be allocated following the "Specification
   Required" policy [RFC5226].

   The Credential Type field of the Server-Trusted-Root TLV Section 6.16
   may assume a value between 0 and 65535, of which 1 is allocated in
   this document.  Additional Credential Type values may be allocated
   following the "Specification Required" policy [RFC5226].

   The Action field field of the Request-Action TLV may assume a value
   between 0 and 65535, of which 0-2 have already been allocated.
   Additional Action values may be allocated following the
   "Specification Required" policy [RFC5226].

9.  Contributors

   A great deal of the text in the first draft of this note was taken
   from a document by Ashwin Palekar, Dan Simon, Glen Zorn, Simon
   Josefsson, Hao Zhou and Joe Salowey; the authors gratefully
   acknowledge their contribution.

   TEAM is a direct descendent of the Protected Extensible
   Authentication Protocol (PEAP), which was created by Glen Zorn while
   employed by Cisco Systems.

10.  Acknowledgements

   Hakan Andersson, Jan-Ove Larsson, Magnus Nystrom, Bernard Aboba,
   Vivek Kamath, Stephen Bensley, Narendra Gidwani, Ilan Frenkel, Nancy
   Cam-Winget, Victor Lortz, Ashwin Palekar, Dan Simon, Glen Zorn, Simon
   Josefsson, Hao Zhou, Joe Salowey, Bernard Aboba, Paul Funk and Jose
   Puthenkulam all contributed at various stages to the development of
   this protocol.

11.  References

11.1.  Normative References

   [I-D.zorn-emu-eap-pwc]        Zorn, G., "A Method for Changing
                                 Cleartext Passwords in the Extensible
                                 Authentication Protocol", March 2011.

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

   [RFC3454]                     Hoffman, P. and M. Blanchet,
                                 "Preparation of Internationalized
                                 Strings ("stringprep")", RFC 3454,
                                 December 2002.

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

   [RFC3986]                     Berners-Lee, T., Fielding, R., and L.
                                 Masinter, "Uniform Resource Identifier
                                 (URI): Generic Syntax", STD 66,
                                 RFC 3986, January 2005.

   [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.

   [RFC5226]                     Narten, T. and H. Alvestrand,
                                 "Guidelines for Writing an IANA
                                 Considerations Section in RFCs",
                                 BCP 26, RFC 5226, May 2008.

    [RFC5246]                     Dierks, T. and E. Rescorla, "The
                                  Transport Layer Security (TLS) Protocol
                                  Version 1.2", RFC 5246, August 2008.

11.2.  Informative References

    [I-D.ietf-emu-eaptunnel-req]  Hoeper, K., Hanna, S., Zhou, H., and J.
                                  Salowey, "Requirements for a Tunnel
                                  Based EAP Method",
                                  draft-ietf-emu-eaptunnel-req-09 (work
                                  in progress), December 2010.

    [IEEE.802-11.2007]            IEEE Computer Society, "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", IEEE Standard 802.11,
                                  June 2007.

    [IEEE.802-1X.2004]            IEEE Computer Society, "IEEE Standard
                                  for Local and metropolitan area
                                  networks: Port-Based Network Access
                                  Control", IEEE Standard 802.1X,
                                  December 2004.

    [RFC1968]                     Meyer, G. and K. Fox, "The PPP
                                  Encryption Control Protocol (ECP)",
                                  RFC 1968, June 1996.

    [RFC1990]                     Sklower, K., Lloyd, B., McGregor, G.,
                                  Carr, D., and T. Coradetti, "The PPP
                                  Multilink Protocol (MP)", RFC 1990,
                                  August 1996.

    [RFC2315]                     Kaliski, B., "PKCS #7: Cryptographic
                                  Message Syntax Version 1.5", RFC 2315,
                                  March 1998.

    [RFC2548]                     Zorn, G., "Microsoft Vendor-specific
                                  RADIUS Attributes", RFC 2548,
                                  March 1999.

    [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.

     [RFC2637]                   Hamzeh, K., Pall, G., Verthein, W.,
                                 Taarud, J., Little, W., and G. Zorn,
                                 "Point-to-Point Tunneling Protocol",
                                 RFC 2637, July 1999.

     [RFC2865]                   Rigney, C., Willens, S., Rubens, A.,
                                 and W. Simpson, "Remote Authentication
                                 Dial In User Service (RADIUS)",
                                 RFC 2865, June 2000.

     [RFC3579]                   Aboba, B. and P. Calhoun, "RADIUS
                                 (Remote Authentication Dial In User
                                 Service) Support For Extensible
                                 Authentication Protocol (EAP)",
                                 RFC 3579, September 2003.

     [RFC3580]                   Congdon, P., Aboba, B., Smith, A.,
                                 Zorn, G., and J. Roese, "IEEE 802.1X
                                 Remote Authentication Dial In User
                                 Service (RADIUS) Usage Guidelines",
                                 RFC 3580, September 2003.

     [RFC3766]                   Orman, H. and P. Hoffman, "Determining
                                 Strengths For Public Keys Used For
                                 Exchanging Symmetric Keys", BCP 86,
                                 RFC 3766, April 2004.

     [RFC3931]                   Lau, J., Townsley, M., and I. Goyret,
                                 "Layer Two Tunneling Protocol - Version
                                 3 (L2TPv3)", RFC 3931, March 2005.

     [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.

   [RFC5216]                        Simon, D., Aboba, B., and R. Hurst,
                                    "The EAP-TLS Authentication Protocol",
                                    RFC 5216, March 2008.

   [RFC5247]                        Aboba, B., Simon, D., and P. Eronen,
                                    "Extensible Authentication Protocol
                                    (EAP) Key Management Framework",
                                    RFC 5247, August 2008.

   [RFC5869]                        Krawczyk, H. and P. Eronen, "HMAC-based
                                    Extract-and-Expand Key Derivation
                                    Function (HKDF)", RFC 5869, May 2010.

   [RFC5931]                        Harkins, D. and G. Zorn, "Extensible
                                    Authentication Protocol (EAP)
                                    Authentication Using Only a Password",
                                    RFC 5931, August 2010.

Appendix A.  Compliance with Requirements for a Tunnel Based EAP Method

   The following subsections describe the TEAM protocol's compliance
   with the requirements given in [I-D.ietf-emu-eaptunnel-req].

A.1.  General Requirements

   o  TEAM includes a Security Claims section and satisfies all
      mandatory requirements listed in section 2.2 of [RFC4017].
   o  TEAM meets the MUST and SHOULD requirements of [RFC5247],
      including generation of the MSK, ESMK, Peer-Id, Server-Id, and
      Session-Id.
   o  TEAM is not tied to any single cryptographic algorithm.  A variety
      of ciphersuites can be negotiated in TEAM phase 1 which include a
      plethora of cryptographic algorithms.  Numerous phase 2
      authentication methods are supported which, likewise, constitute a
      plethora of cryptographic algorithms.
   o  TEAM meets all the MUST and SHOULD requirements in section 3 of
      [RFC4962] to the extent that they apply to an EAP method (and not
      to the use of a key derived from TEAM).  In particular TEAM keys
      are vine-ripened and very fresh.

A.2.  Tunnel Requirements

   o  TEAM uses TLS version 1.2 in phase 1 and satisfies all of the
      mandatory TLS requirements of section 4.2.1
      [I-D.ietf-emu-eaptunnel-req].
   o  TEAM supports fragmentation and reassembly per section 4.2.2 of
      [I-D.ietf-emu-eaptunnel-req].

   o  Modification of data outside the tunnel is not protected but any
      such modification does not cause an exploitable vulnerability and
      can be detected Section 7.1.

A.3.  Tunnel Payload Requirements

   o  TEAM AVPs are extensible.
   o  TEAM is an EAP method and supports challenge/response operations
      that are typical of EAP methods.
   o  It is possible to indicate whether a TLV is mandatory or not.
   o  TEAM supports Vendor Specific extensions.
   o  TEAM supports indication of result after each changed inner
      method.

A.4.  Channel Binding Requirements

   o  To the extent that it is appropriate to rely on adherence to a
      "work-in-progress", TEAM supports Channel Binding requirements.
      Furthermore, as that "work-in-progress" proceeds in its work there
      is no reason why TEAM could not continue to meet requirements.

A.5.  Username/Password Requirements

   o  TEAM supports the required use of usernames and passwords in
      section 4.5 of [I-D.ietf-emu-eaptunnel-req] through the use of the
      EAP/Identity exchange and GTC [RFC3748], and EAP-PWC
      [I-D.zorn-emu-eap-pwc].  Note, however, that in order to comply
      with the requirements of [I-D.ietf-emu-eaptunnel-req] the user
      name contained in the EAP/Identity/Response and the password
      contined in the EAP-GTC/Response messages MUST be processed
      according to the rules of the [RFC4013] profile of [RFC3454].
      Furthermore, the strings in question SHALL be considered to be
      "stored strings" (per [RFC3454]), and unassigned code points are
      therefore prohibited.  The output SHALL be the binary
      representation of the processed UTF-8 character string.
      Prohibited output and unassigned codepoints encountered during
      SASLprep pre-processing SHALL cause a failure of pre-processing,
      and the output MUST NOT be used.
   o  In addition, TEAM supports the use of username/password
      authentication that allows for an EAP peer and EAP server to
      authenticate each other based on knowledge of a password without
      that password being sent in any format between the peer and
      server.
   o  The TEAM server is authenticated before any possible transmission
      of a password and a peer can check whether the certificate of a
      TEAM server has been revoked or not using OCSP.

A.6.  Requirements Around Carriage of EAP Methods

     o  TEAM supports carrying inner EAP methods without modification.
        These methods are negotiated and can be chained.
     o  TEAM supports cryptographic binding of keys derived from EAP
        methods.

Authors' Addresses

     Glen Zorn
     Network Zen
     227/358 Thanon Sanphawut
     Bang Na, Bangkok  10260
     Thailand

     Phone: +66 (0) 87-040-4617
     EMail: gwz@net-zen.net


     Qin Wu
     Huawei Technologies Co., Ltd.
     101 Software Avenue, Yuhua District
     Nanjing, Jiangsu  21001
     China

     Phone: +86-25-84565892
     EMail: sunseawq@huawei.com


     Dan Harkins
     Aruba Networks
     1322 Crossman Avenue
     Sunnyvale, CA  94089-1113
     United States of America

     EMail: dharkins@arubanetworks.com