EAP
Internet-Draft
Category: Standards Track
<draft-funk-eap-ttls-v1-01.txt>

Paul Funk
Juniper Networks
Simon Blake-Wilson
Basic Commerce &
Industries, Inc.
March 2006

EAP Tunneled TLS Authentication Protocol Version 1 (EAP-TTLSv1)

Status of this Memo

Copyright Notice

Abstract

EAP-TTLS is an EAP type that utilizes TLS to establish a secure connection between a client and server, through which additional information may be exchanged. The initial TLS handshake may mutually authenticate client and server; or it may perform a one-way authentication, in which only the server is authenticated to the client. The secure connection established by the initial handshake may then be used to allow the server to authenticate the client using existing, widely-deployed authentication infrastructures such

as RADIUS. The authentication of the client may itself be EAP, or it may be another authentication protocol such as PAP, CHAP, MS-CHAP or MS-CHAP-V2.

Thus, EAP-TTLS allows legacy password-based authentication protocols to be used against existing authentication databases, while protecting the security of these legacy protocols against eavesdropping, man-in-the-middle and other cryptographic attacks.

EAP-TTLS also allows client and server to exchange other information in addition to authentication-related information.

This document describes EAP-TTLSv1; that is, version 1 of the EAP-TTLS protocol. It represents a significant enhancement to the original version 0 of the protocol. EAP-TTLSv1 utilizes an extended version of TLS, called TLS/IA (TLS/InnerApplication) as its underlying protocol [TLS/IA].

Table of Contents

1. Introduction

EAP-TTLS is an EAP type that utilizes TLS to establish a secure connection between a client and server, through which additional information may be exchanged. The initial TLS handshake may mutually authenticate client and server; or it may perform a one-way authentication, in which only the server is authenticated to the client. The secure connection established by the initial handshake may then be used to allow the server to authenticate the client using existing, widely-deployed authentication infrastructures such as RADIUS. The authentication of the client may itself be EAP, or it may be another authentication protocol such as PAP, CHAP, MS-CHAP or MS-CHAP-V2.

Thus, EAP-TTLS allows legacy password-based authentication protocols to be used against existing authentication databases, while protecting the security of these legacy protocols against eavesdropping, man-in-the-middle and other cryptographic attacks.

EAP-TTLS also allows client and server to establish keying material for use in the data connection between the client and access point. The keying material is established implicitly between client and server based on the TLS handshake.

This document describes EAP-TTLSv1; that is, version 1 of the EAP-TTLS protocol. It represents a significant enhancement to the original version 0 of the protocol. (EAP-TTLSv0).

1.1 EAP-TTLSv1

EAP-TTLSv1 utilizes TLS with the Inner Application extension (TLS/IA), as its underlying protocol. In TLS/IA, the TLS handshake is followed by an exchange of messages with record type InnerApplication, in which an arbirary exchange of messages between client and server is conducted under the confidentiality and integrity protection afforded by the TLS handshake.

The InnerApplication messages that are exchanged between client and server are encoded as sequences of Attribute-Value-Pairs (AVPs) from the RADIUS/Diameter namespace. Use of the RADIUS/Diameter namespace provides natural compatibility between TLS/IA applications and widely deployed AAA infrastructures. This namespace is extensible, allowing new AVPs and, thus, new applications to be defined as needed, either by standards bodies or by vendors wishing to define proprietary applications.

The AVPs exchanged between client and server typically provide for client authentication, or mutual client-server authentication. However, the AVP exchange accommodates any type of client-server

exchange, not just authentication, though authentication may often be the prerequisite that allows other exchanges to proceed. For

example, EAP-TTLSv1 may be used to verify endpoint integrity, provision keying material for use in separate data channel communications (e.g. IPsec), provide client credentials for single sign-on, and so on.

1.2 Differences From Version 0

Version 1 of EAP-TTLS is similar to version 0 in that a TLS handshake is used to protect a subsequent AVP exchange. In version 0, the handshake portion of TLS is used to establish a tunnel and the data portion is used to carry AVPs. This approach is similar to that of other tunneled protocols, such as EAP-PEAP and EAP-FAST.

In version 1, an extension to TLS, called TLS/IA, is utilized; TLS/IA already provides for a protected AVP exchange following the TLS handshake, in effect producing an "extended" handshake. TLS/IA was developed to allow authentication and other client-server negotiations to occur within TLS itself. Thus, TLS/IA is suitable both as the underlying protocol for EAP methods as well as a means of introducing authentication and other client-server exchanges when TLS is used to protect data communications such as an HTTP conversation.

Use of TLS/IA in version 1 of EAP-TTLS provides several improvements over verion 0:

- Inner authentications are confirmed by mixing session keys developed from those authentications with the master secret developed during the TLS handshake. This guarantees that the TLS handshake endpoint and the authentication endpoint are one and the same, thus eliminating the Man-in-the-Middle (MitM) attack against tunneled protocols for inner authentications that generate session keys. See [MITM] for information about this attack.

- Session keys developed from inner authentications are mixed with the TLS master secret to produce an "inner secret", which is exported by TLS/IA. The inner secret is used to generate the MSK (master session key) exported by EAP-TTLSv1 for use in the subsequent data connection. Use of a session key that is bound to inner session keys guarantees that the subsequent data connection wll not operate except with the authentic client, even if the original TLS master secret were compromised and available to an eavesdropper.

- TLS/IA's multi-phase operation allows a subsequent phase to confirm the results of prior phases before proceeding.

- A secure final exchange of the result of inner authentication is

exchanged between client and server to conclude the EAP-TTLSv1 exchange. This precludes any possibility of truncation attack

that could occur when the client relies solely on an unprotected EAP-Success message to determine that the server has completed its authentication.

2. Motivation

Most password-based protocols in use today rely on a hash of the password with a random challenge. Thus, the server issues a challenge, the client hashes that challenge with the password and forwards a response to the server, and the server validates that response against the user's password retrieved from its database. This general approach describes CHAP, MS-CHAP, MS-CHAP-V2, EAP/MD5-Challenge and EAP/One-Time Password.

An issue with such an approach is that an eavesdropper that observes both challenge and response may be able to mount a dictionary attack, in which random passwords are tested against the known challenge to attempt to find one which results in the known response. Because passwords typically have low entropy, such attacks can in practice easily discover many passwords.

While this vulnerability has long been understood, it has not been of great concern in environments where eavesdropping attacks are unlikely in practice. For example, users with wired or dial-up connections to their service providers have not been concerned that such connections may be monitored. Users have also been willing to entrust their passwords to their service providers, or at least to allow their service providers to view challenges and hashed responses which are then forwarded to their home authentication servers using, for example, proxy RADIUS, without fear that the service provider will mount dictionary attacks on the observed credentials. Because a user typically has a relationship with a single service provider, such trust is entirely manageable.

With the advent of wireless connectivity, however, the situation changes dramatically:

- Wireless connections are considerably more susceptible to eavesdropping and man-in-the-middle attacks. These attacks may enable dictionary attacks against low-entropy passwords. In addition, they may enable channel hijacking, in which an attacker gains fraudulent access by seizing control of the communications channel after authentication is complete.

- Existing authentication protocols often begin by exchanging the client s username in the clear. In the context of eavesdropping on the wireless channel, this can compromise the client s anonymity and locational privacy.

- Often in wireless networks, the access point does not reside in the administrative domain of the service provider with which the

user has a relationship. For example, the access point may reside in an airport, coffee shop, or hotel in order to provide public access via 802.11. Even if password authentications are protected in the wireless leg, they may still be susceptible to eavesdropping within the untrusted wired network of the access point.

- In the traditional wired world, the user typically intentionally connects with a particular service provider by dialing an associated phone number; that service provider may be required to route an authentication to the user's home domain. In a wireless network, however, the user does not get to choose an access domain, and must connect with whichever access point is nearby; providing for the routing of the authentication from an arbitrary access point to the user's home domain may pose a challenge.

Thus, the authentication requirements for a wireless environment that EAP-TTLS attempts to address can be summarized as follows:

- Legacy password protocols must be supported, to allow easy deployment against existing authentication databases.

- Password-based information must not be observable in the communications channel between the client node and a trusted service provider, to protect the user against dictionary attacks.

- The user's identity must not be observable in the communications channel between the client node and a trusted service provider, to protect the user's locational privacy against surveillance, undesired acquisition of marketing information, and the like.

- The authentication process must result in the distribution of shared keying information to the client and access point to permit encryption and validation of the wireless data connection subsequent to authentication, to secure it against eavesdroppers and prevent channel hijacking.

- The authentication mechanism must support roaming among small access domains with which the user has no relationship and which will have limited capabilities for routing authentication requests.

3. Terminology

AAA

Authentication, Authorization and Accounting - functions that are generally required to control access to a network and support billing and auditing.

A network protocol used to communicate with AAA servers; examples include RADIUS and Diameter.

AAA server

A server which performs one or more AAA functions: authenticating a user prior to granting network service, providing authorization (policy) information governing the type of network service the user is to be granted, and accumulating accounting information about actual usage.

AAA/H

A AAA server in the user's home domain, where authentication and authorization for that user are administered.

access point

A network device providing users with a point of entry into the network, and which may enforce access control and policy based on information returned by a AAA server. For the purposes of this document, "access point" and "NAS" are architecturally equivalent. "Access point" is used throughout because it is suggestive of devices used for wireless access; "NAS" is used when more traditional forms of access, such as dial-up, are discussed.

access domain

The domain, including access points and other devices, that provides users with an initial point of entry into the network; for example, a wireless hot spot.

client

A host or device that connects to a network through an access point.

domain

A network and associated devices that are under the administrative control of an entity such as a service provider or the user's home organization.

link layer protocol

A protocol used to carry data between hosts that are connected within a single network segment; examples include PPP and Ethernet.

A Network Access Identifier [RFC2486], normally consisting of the name of the user and, optionally, the user's home realm.

NAS

A network device providing users with a point of entry into the network, and which may enforce access control and policy based on information returned by a AAA server. For the purposes of this document, "access point" and "NAS" are architecturally equivalent. "Access point" is used throughout because it is suggestive of devices used for wireless access; "NAS" is used when more traditional forms of access, such as dial-up, are discussed.

proxy

A server that is able to route AAA transactions to the appropriate AAA server, possibly in another domain, typically based on the realm portion of an NAI.

realm

The optional part of an NAI indicating the domain to which a AAA transaction is to be routed, normally the user's home domain.

service provider

An organization with which a user has a business relationship, that provides network or other services. The service provider may provide the access equipment with which the user connects, may perform authentication or other AAA functions, may proxy AAA transactions to the user's home domain, etc.

TTLS server

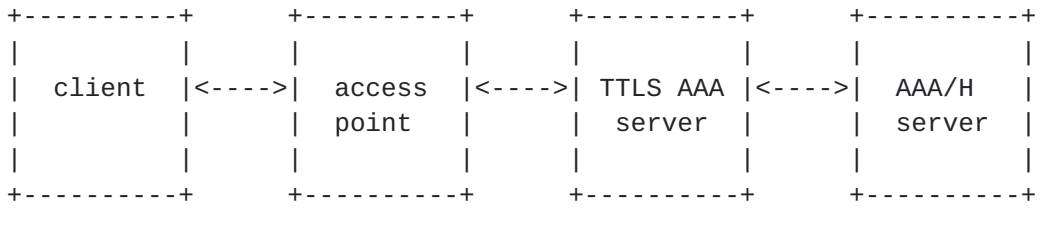
A AAA server which implements EAP-TTLS. This server may also be capable of performing user authentication, or it may proxy the user authentication to a AAA/H.

user

The person operating the client device. Though the line is often blurred, "user" is intended to refer to the human being who possesses an identity (username), password or other authenticating information, and "client" is intended to refer to the device which makes use of this information to negotiate network access. There may also be clients with no human operators; in this case the term "user" is a convenient abstraction.

4. Architectural Model

The network architectural model for EAP-TTLS usage and the type of security it provides is shown below.



The entities depicted above are logical entities and may or may not correspond to separate network components. For example, the TTLS server and AAA/H server might be a single entity; the access point and TTLS server might be a single entity; or, indeed, the functions of the access point, TTLS server and AAA/H server might be combined into a single physical device. The above diagram illustrates the division of labor among entities in a general manner and shows how a distributed system might be constructed; however, actual systems might be realized more simply.

Note also that one or more AAA proxy servers might be deployed between access point and TTLS server, or between TTLS server and AAA/H server. Such proxies typically perform aggregation or are required for realm-based message routing. However, such servers play no direct role in EAP-TTLS and are therefore not shown.

4.1 Carrier Protocols

The entities shown above communicate with each other using carrier protocols capable of encapsulating EAP. The client and access point communicate using a link layer carrier protocol such as PPP or EAPOL. The access point, TTLS server and AAA/H server communicate using a AAA carrier protocol such as RADIUS or Diameter.

EAP, and therefore EAP-TTLS, must be initiated via the link layer protocol. In PPP or EAPOL, for example, EAP is initiated when the access point sends an EAP-Request/Identity packet to the client.

The keying material used to encrypt and authenticate the data connection between the client and access point is developed implicitly between the client and TTLS server as a result of EAP-TTLS negotiation. This keying material must be communicated to the access point by the TTLS server using the AAA carrier protocol.

The client and access point must also agree on an encryption/validation algorithm to be used based on the keying material. In some systems, both these devices may be preconfigured with this information, and distribution of the keying material alone is sufficient. Or, the link layer protocol may provide a mechanism for client and access point to negotiate an algorithm.

In the most general case, however, it may be necessary for both client and access point to communicate their algorithm preferences to the TTLS server, and for the TTLS server to select one and communicate its choice to both parties. This information would be transported between access point and TTLS server via the AAA protocol, and between client and TTLS server via EAP-TTLS in encrypted form.

4.2 Security Relationships

The client and access point have no pre-existing security relationship.

The access point, TTLS server and AAA/H server are each assumed to have a pre-existing security association with the adjacent entity with which it communicates. With RADIUS, for example, this is achieved using shared secrets. It is essential for such security relationships to permit secure key distribution.

The client and AAA/H server have a security relationship based on the user's credentials such as a password.

The client and TTLS server may have a one-way security relationship based on the TTLS server's possession of a private key guaranteed by a CA certificate which the user trusts, or may have a mutual security relationship based on certificates for both parties.

4.3 Messaging

The client and access point initiate an EAP conversation to negotiate the client's access to the network. Typically, the access point issues an EAP-Request/Identity to the client, which responds with an EAP-Response/Identity. Note that the client does not include the user's actual identity in this EAP-Response/Identity packet; the user's identity will not be transmitted until an encrypted channel has been established.

The access point now acts as a passthrough device, allowing the TTLS server to negotiate EAP-TTLS with the client directly.

During the first phase of the negotiation, the TLS handshake protocol is used to authenticate the TTLS server to the client and,

optionally, to authenticate the client to the TTLS server, based on public/private key certificates. As a result of the handshake,

client and TTLS server now have shared keying material and an agreed upon TLS record layer cipher suite with which to secure subsequent EAP-TTLS communication.

During the second phase of negotiation, client and TTLS server use the secure TLS record layer channel established by the TLS handshake as a tunnel to exchange information encapsulated in attribute-value pairs, to perform additional functions such as client authentication and key distribution for the subsequent data connection.

If a tunneled client authentication is performed, the TTLS server de-tunnels and forwards the authentication information to the AAA/H. If the AAA/H performs a challenge, the TTLS server tunnels the challenge information to the client. The AAA/H server may be a legacy device and needs to know nothing about EAP-TTLS; it only needs to be able to authenticate the client based on commonly used authentication protocols.

Keying material for the subsequent data connection between client and access point may be generated based on secret information developed during the TLS handshake and subsequent tunneled authentications between client and TTLS server. At the conclusion of a successful authentication, the TTLS server may transmit this keying material to the access point, encrypted based on the existing security associations between those devices (e.g., RADIUS).

The client and access point now share keying material which they can use to encrypt data traffic between them.

In EAP-TTLSv1, the AVP exchange during the second phase is performed using InnerApplication records via the TLS/IA protocol. This AVP exchange itself may be be multi-phase, with each phase proceeding only if the prior phase resulted in success.

4.4 Resulting Security

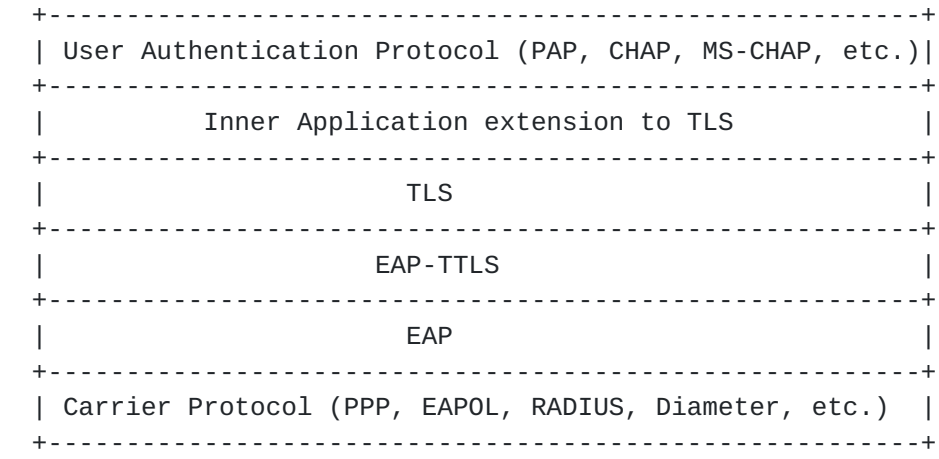
As the diagram above indicates, EAP-TTLS allows user identity and password information to be securely transmitted between client and TTLS server, and performs key distribution to allow network data subsequent to authentication to be securely transmitted between client and access point.

5. Protocol Layering Model

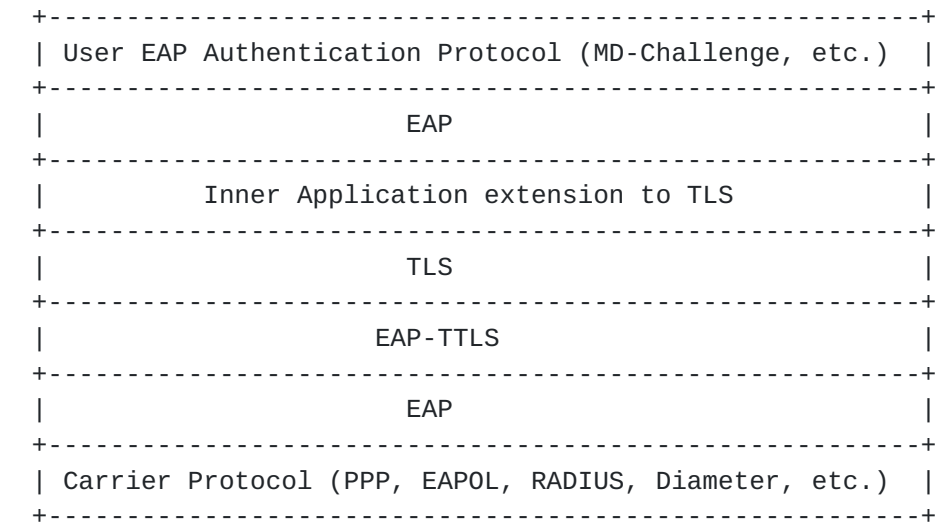
EAP-TTLSv1 packets are encapsulated within EAP, and EAP in turn requires a carrier protocol to transport it. EAP-TTLSv1 packets themselves encapsulate TLS/IA, which is then used to encapsulate user authentication information. TLS/IA, as an extension to TLS, can be considered encapsulated by TLS. Thus, EAP-TTLSv1 messaging can be

described using a layered model, where each layer is encapsulated by

the layer beneath it. The following diagram clarifies the relationship between protocols:



When the user authentication protocol is itself EAP, the layering is as follows:



Methods for encapsulating EAP within carrier protocols are already defined. For example, PPP [RFC1661] or EAPOL [802.1X] may be used to transport EAP between client and access point; RADIUS [RFC2685] or Diameter [RFC3588] are used to transport EAP between access point and TTLS server.

6. EAP-TTLSv1 Overview

EAP-TTLSv1 is initiated by the server's transmission of a Start packet to the client.

The EAP exchange proceeds with transmission of TLS/IA message

sequences alternately by client and server, with each message sequence encapsulated in an EAP-TTLSv1 frame. Descriptions of the TLS/IA messages can be found in [TLS/IA].

A successful authentication will result in the server sending a TLS/IA FinalPhaseFinished message and the client responding with it's own FinalPhaseFinished message.

The server then sends an EAP-Success to the client to complete the authentication. This message is the standard EAP success message and is sent in the clear.

Client and server each computes the MSK (the Master Sesion Key, as defined in [RFC3784]), based on information generated in the TLS/IA exchange. The server may then transmit the MSK to the access point for use in its data communications with the client.

If the TLS/IA negotiation fails, the server sends an EAP-Failure to the client.

6.1 Session Resumption

When a client and TTLS server that have previously negotiated a EAP-TTLSv1 session begin a new EAP-TTLSv1 negotiation, the client and TTLS server may agree to resume the previous session. This significantly reduces the time required to establish the new session. This could occur when the client connects to a new access point, or when an access point requires reauthentication of a connected client.

Session resumption is accomplished using the standard TLS mechanism. The client signals its desire to resume a session by including the session ID of the session it wishes to resume in the ClientHello message; the TTLS server signals its willingness to resume that session by echoing that session ID in its ServerHello message.

If the TTLS server elects not to resume the session, it simply does not echo the session ID and a new session will be negotiated. This could occur if the TTLS server is configured not to resume sessions, if it has not retained the requested session's state, or if the session is considered stale. A TTLS server may consider the session stale based on its own configuration, or based on session-limiting information received from the AAA/H (e.g., the RADIUS Session-Timeout attribute).

Addition messages beyond the TLS handshake may or may not occur within a resumed session. TLS/IA provides a negotiation mechanism allowing client and server to determine whether InnerApplication messages are to ensue upon session resumption. Typically, inner authentications would not be required in a resumed session, as the ability to resume the session may provide sufficient evidence to either party of the identity of the other. However, there may be additional information that needs to be refreshed or renegotiated

during a session resumption.

When an inner authentication is not performed during a resumed session, the TTLS server will not receive new authorization information from the AAA/H. In this case, the TTLS server must retain authorization information initially returned by the AAA/H for use in resumed sessions. Authorization information might include the maximum time for the session, the maximum allowed bandwidth, packet filter information and the like. The TTLS server is responsible for modifying time values, such as Session-Timeout, appropriately for each resumed session.

A TTLS server MUST NOT permit a session to be resumed if that session did not result in a successful completion of the entire TLS/IA exchange, and a client MUST NOT propose the session ID of a failed session for resumption. The consequence of incorrectly implementing this aspect of session resumption would be catastrophic; any attacker could easily gain network access by first initiating a session that succeeds in the TLS handshake but fails the inner authentication, and then resuming that session.

[Implementation note: Toolkits that implement TLS often cache resumable TLS sessions automatically. Implementers must take care to override such automatic behavior, and prevent sessions from being cached for possible resumption until the user has been positively authenticated.]

A TTLS server MUST NOT permit a session negotiated with different tunneled TLS-based EAP protocol to be resumed in an EAP-TTLSv1 session, and a client MUST NOT propose the session ID resulting from such a protocol for resumption in EAP-TTLSv1. Note that previous versions of EAP-TTLS are considered different tunneled TLS-based protocols for the purposes of this paragraph. Thus, a session negotiated using EAP-PEAP, EAP-FAST or EAP-TTLSv0 are not candidate sessions for resumption in EAP-TTLSv1.

6.1.1 TTLS Server Guidelines for Session Resumption

When a domain comprises multiple TTLS servers, a client's attempt to resume a session may fail because each EAP-TTLS negotiation may be routed to a different TTLS server.

One strategy to ensure that subsequent EAP-TTLS negotiations are routed to the original TTLS server is for each TTLS server to encode its own identifying information, for example, IP address, in the session IDs that it generates. This would allow any TTLS server receiving a session resumption request to forward the request to the TTLS server that established the original session.

7. Generating Keying Material

Upon successful conclusion of an EAP-TTLSv1 negotiation, a 64-octet MSK (Master Session Key) is generated and exported for use in securing the data connection between client and access point.

The MSK is generated using the TLS PRF function [RFC2246], with inputs consisting of the inner secret exported by TLS/IA, the ASCII-encoded constant string "ttls v1 keying material", the TLS client random, and the TLS server random. The constant string is not null-terminated. The TLS/IA inner secret, rather than the TLS master secret, is used because it binds session keys from inner authentications with the TLS master secret and therefore provides greater security in the (unlikely) case that an adversary is able to compromise the master secret.

```
MSK = PRF(inner_secret,
          "ttls v1 keying material",
          SecurityParameters.client_random +
          SecurityParameters.server_random) [0..63]
```

Note that the order of client_random and server_random for EAP-TTLS is reversed from that of the TLS protocol [RFC2246]. This ordering follows the key derivation method of EAP-TLS [RFC2716]. Altering the order of randoms avoids namespace collisions between constant strings defined for EAP-TTLSv1 and those defined for the TLS protocol.

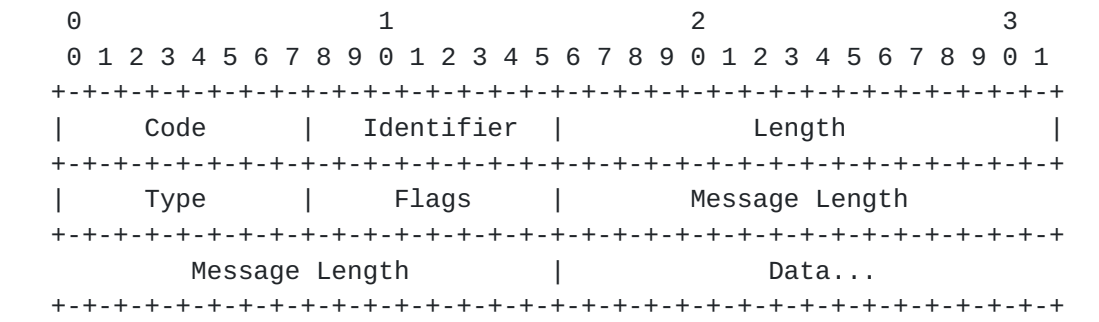
The inner secret used in the PRF MUST be the one generated at the conclusion of the final InnerApplication phase of TLS/IA; the client random and server random MUST be those established during the TLS handshake. Client and TTLS server generate this keying material independently, and the result is guaranteed to be the same for each if the TLS/IA exchange succeeds.

The TTLS server distributes this keying material to the access point via the AAA carrier protocol. When RADIUS is the AAA carrier protocol, the MPPE-Recv-Key and MPPE-Send-Key attributes may be used to distribute the first 32 octets and second 32 octets of the MSK, respectively.

8. EAP-TTLSv1 Protocol

8.1 Packet Format

The EAP-TTLSv1 packet format is shown below. The fields are transmitted left to right.



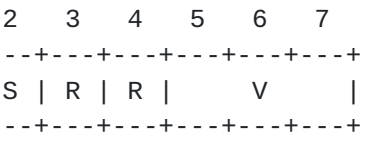
Code
1 for request, 2 for response.

Identifier
The Identifier field is one octet and aids in matching responses with requests. The Identifier field MUST be changed for each request packet and MUST be echoed in each response packet.

Length
The Length field is two octets and indicates the number of octets in the entire EAP packet, from the Code field through the Data field.

Type
21 (EAP-TTLS, all versions)

Flags



L = Length included
M = More fragments
S = Start
R = Reserved
V = Version (001 for EAP-TTLSv1)

The L bit is set to indicate the presence of the four octet TLS Message Length field. The M bit indicates that more fragments are to come. The S bit indicates a Start message. The V bit is set to the version of EAP-TTLS, and is set to 001 for EAP-TTLSv1.

Message Length
The Message Length field is four octets, and is present only if the L bit is set. This field provides the total length of the raw data message sequence prior to fragmentation.

Data

For all packets other than a Start packet, the Data field consists of the raw TLS message sequence or fragment thereof. For

a Start packet, the Data field may optionally contain an AVP sequence.

8.2 EAP-TTLS Start Packet

The S bit MUST be set on the first packet sent by the server to initiate the EAP-TTLSv1 protocol. It MUST NOT be set on any other packet.

This packet MAY contain additional information in the form of AVPs, which may provide useful hints to the client; for example, the server identity may be useful to the client to allow it to pick the correct TLS session ID for session resumption. Each AVP must begin on a 4-octet boundary relative to the first AVP in the sequence. If an AVP is not a multiple of 4 octets, it must be padded with 0s to the next 4-octet boundary.

8.2.1 Version Negotiation

The version of EAP-TTLS is negotiated in the first exchange between server and client. The server sets the highest version number of EAP-TTLS that it supports in the V field of its Start message (in the case of EAP-TTLS v1, this is 1). In its first EAP message in response, the client sets the V field to the highest version number that it supports that is no higher than the version number offered by the server. If the client version is not acceptable to the server, it sends an EAP-Failure to terminate the EAP session. Otherwise, the version sent by the client is the version of EAP-TTLS that MUST be used, and both server and client set the V field to that version number in all subsequent EAP messages.

8.2.2 Fragmentation

Each EAP-TTLSv1 message contains a sequence of TLS messages that represent a single leg of a half-duplex conversation. The EAP carrier protocol (e.g., PPP, EAPOL, RADIUS) may impose constraints on the length of of an EAP message. Therefore it may be necessary to fragment an EAP-TTLSv1 message across multiple EAP messages.

Each fragment except for the last MUST have the M bit set, to indicate that more data is to follow; the final fragment MUST NOT have the M bit set.

If there are multiple fragments, the first fragment MUST have the L bit set and include the length of the entire raw message prior to fragmentation. Fragments other than the first MUST NOT have the L bit set.

Upon receipt of a packet with M bit set, the receiver MUST transmit

an Acknowledgement packet. The receiver is responsible for reassembly of fragmented packets.

8.2.3 Acknowledgement Packets

An Acknowledgement packet is an EAP-TTLSv1 packet with no additional data beyond the Flags octet, and with the L, M and S bits of the Flags octet set to 0. (Note, however, that the V field MUST still be set to the appropriate version number.)

An Acknowledgement packet is sent for the following purposes:

- Fragment Acknowledgement

  A Fragment Acknowledgement is sent in response to an EAP packet with M bit set.

- Error Alert Acknowledgement

  Either party may at any time send a TLS error alert to fail the TLS/IA handshake.

  If the client sends an error alert to the server, no further EAP-TTLS messages are exchanged, and the server sends an EAP-Failure to terminate the conversation.

  If the server sends an error alert to the client, the client MUST respond with an Acknowledgement packet to allow the conversation to continue. Upon receipt of the Acknowledgement packet, the server sends an EAP-Failure to terminate the conversation.

Note that, unlike EAP-TTLSv0, in EAP-TTLSv1 there is no case in which a client sends a packet with data as a result of having no AVPs to send. In EAP-TTLSv1, if no AVPs are to be sent, there will nevertheless be an InnerApplication message carrying zero AVPs, which the client must send.

9. Security Claims

Pursuant to [RFC3748], security claims for EAP-TTLSv1 are as follows:

```
Authentication mechanism: TLS plus arbitrary additional protected
                           authentication(s)
Ciphersuite negotiation:  Yes
Mutual authentication:    Yes, in recommended implementation
Integrity protection:     Yes
Replay protection:        Yes
Confidentiality:          Yes
Key derivation:           Yes
Key strength:             384 bits or higher
Dictionary attack prot.:  Yes
```

Fast reconnect:	Yes
Crypt. binding:	Yes

Session independence:	Yes
Fragmentation:	Yes
Channel binding:	Supported via AVPs, though optional

10. Security Considerations

This draft is entirely about security and the security considerations associated with the mechanisms employed in this document should be considered by implementers.

The following additional issues are relevant:

- Anonymity and privacy. EAP-TTLS does not communicate a username in the clear in the initial EAP-Response/Identity. This feature is designed to support anonymity and location privacy from attackers eavesdropping the network path between the client and the TTLS server. However implementers should be aware that other factors - both within EAP-TTLS and elsewhere - may compromise a user's identity. For example, if a user authenticates with a certificate, the subject name in the certificate may reveal the user's identity. Outside of EAP-TTLS, the client's fixed MAC address, or in the case of wireless connections, the client's radio signature, may also reveal information. Additionally, implementers should be aware that a user's identity is not hidden from the TTLS server and may be included in the clear in AAA messages between the TTLS server and the AAA/H server.

- Trust in the TTLS server. EAP-TTLS is designed to allow the use of legacy authentication methods to be extended to mediums like wireless in which eavesdropping the link between the client and the access point is easy. However implementers should be aware of the possibility of attacks by rogue TTLS servers - for example in the event that the inner authentication method is susceptible to dictionary attacks. Therefore it is essential that clients be properly configured to only proceed with inner authentications with trusted TTLS servers, as evidenced by the certificate chain presented by the TTLS server in the TTLS handshake. In general, cipher suites that allow the TTLS server to remain anonymous should be avoided, unless the inner authentication itself provides mutual authentication and is resistant to dictionary attack.

- TTLS server certificate compromise. The use of TTLS server certificates within EAP-TTLS makes EAP-TTLS susceptible to attack in the event that a TTLS server's certificate is compromised. - TTLS servers should therefore take care to protect their private key. In addition, certificate revocation methods may be used to mitigate against the possibility of key compromise. [RFC3546]

describes a way to integrate one such method - OCSP [RFC2560] - into the TLS handshake - use of this approach may be appropriate within EAP-TTLS.

- Listing of data cipher preferences. EAP-TTLS negotiates data cipher suites by having the EAP-TTLS server select the first cipher suite appearing on the client list that also appears on the access point list. In order to maximize security, it is therefore recommended that the client order its list according to security - most secure acceptable cipher suite first, least secure acceptable cipher suite last.

- Forward secrecy. With forward secrecy, revelation of a secret does not compromise session keys previously negotiated based on that secret. Thus, when the TLS key exchange algorithm provides forward secrecy, if a TTLS server certificate's private key is eventually stolen or cracked, tunneled user password information will remain secure as long as that certificate is no longer in use. Diffie-Hellman key exchange is an example of an algorithm that provides forward secrecy. A forward secrecy algorithm should be considered if attacks against recorded authentication or data sessions are considered to pose a significant threat.

11. References

11.1 Normative References

[TLS/IA] Funk, P., Blake-Wilson, S., Smith, N., Tschofenig, H. and T. Hardjono, " TLS Inner Application Extension (TLS/IA)", draft-funk-tls-inner-application-extension-02.txt, March 2006.

[RFC1700] Reynolds, J., and J. Postel, "Assigned Numbers", RFC 1700, October 1994.

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

[RFC2246] Dierks, T., and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, November 1998.

[RFC2486] Aboba, B., and M. Beadles, "The Network Access Identifier", RFC 2486, January 1999.

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

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

[RFC3546] Blake-Wilson, S., Nystrom, M., Hopwood, D., Mikkelsen, J., and T. Wright, "Transport Layer Security (TLS)

Extensions", RFC 3546, June 2003.

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

[RFC3588] Calhoun, P., Loughney, J., Guttman, E., Zorn, G., and J. Arkko, "Diameter Base Protocol", RFC 3588, July 2003.

[RFC3784] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, "PPP Extensible Authentication Protocol (EAP)", RFC 3784, June 2004.

11.2 Informative References

[RFC1661] Simpson, W. (Editor), "The Point-to-Point Protocol (PPP)", STD 51, RFC 1661, July 1994.

[RFC1994] Simpson, W., "PPP Challenge Handshake Authentication Protocol (CHAP)", RFC 1994, August 1996.

[RFC2716] Aboba, B., and D. Simon, "PPP EAP TLS Authentication Protocol", RFC 2716, October 1999.

[RFC2433] Zorn, G., and S. Cobb, "Microsoft PPP CHAP Extensions", RFC 2433, October 1998.

[RFC2560] Myers, M., Ankney, R., Malpani, A., Galperin, S., and C. Adams, "Internet X.509 Public Key Infrastructure: Online Certificate Status Protocol - OCSP", RFC 2560, June 1999.

[RFC2759] Zorn, G., "Microsoft PPP CHAP Extensions, Version 2", RFC 2759, January 2000.

[EAP-PEAP] Palekar, A., Simon, D., Salowey, J., Zhou, H., Zorn, G., and S. Josefsson, "Protected EAP Protocol (PEAP) Version 2", draft-josefsson-pppext-eap-tls-eap-08.txt, July 2004.

[TLS-PSK] Eronen, P., and H. Tschofenig, "Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)", draft-ietf-tls-psk-01.txt, August 2004.

[802.1X] IEEE Standards for Local and Metropolitan Area Networks: Port based Network Access Control, IEEE Std 802.1X-2001, June 2001.

[MITM] Asokan, N., Niemi, V., and K. Nyberg, "Man-in-the-Middle in Tunneled Authentication",

http://www.saunalahti.fi/~asokan/research/mitm.html,
Nokia Research Center, Finland, October 24 2002.

[KEYING] Aboba, B., Simon, D., Arkko, J. and H. Levkowetz, "EAP Key Management Framework", draft-ietf-eap-keying-01.txt (work in progress), October 2003.

[IKEv2] C.Kaufman, "Internet Key Exchange (IKEv2) Protocol", draft-ietf-ipsec-ikev2-16.txt (work in progress), September 2004.

[AAA-EAP] Eronen, P., Hiller, T. and G. Zorn, "Diameter Extensible Authentication Protocol (EAP) Application", draft-ietf-aaa-eap-03.txt (work in progress), October 2003.

12. Authors' Addresses

Questions about this memo can be directed to:

Paul Funk
Juniper Networks
222 Third Street
Cambridge, MA 02142
USA
Phone: +1 617 497-6339
E-mail: pfunk@juniper.net

Simon Blake-Wilson
Basic Commerce & Industries, Inc.
304 Harper Drive, Suite 203
Moorestown, NJ 08057
Phone: +1 856 778-1660
E-mail: sblakewilson@bcisse.com

Disclaimer of Validity

Copyright Statement