**EAP-TTLS**

EAP-TTLSv1 Internet Draft:  
http://tools.ietf.org/html/draft-funk-eap-ttls-v1-01

EAP-TLS RFC (for the EAP-TLS handshake procedure):  
http://tools.ietf.org/html/rfc5216

EAP-TTLSv0 RFC (for completeness):  
https://tools.ietf.org/html/rfc5281

Actors

*Client* - A client trying to authenticate / get access to network resources

*Network Access Server* - Access point to the network (a switch, wireless access point, or something completely different like an SSH server).

*TTLS AAA server* - The server which securely (over EAP/TTLS) negotiates the authentication process for the client

*AAA/H Server* - The AAA (home) server with access to the clients credentials. This can be the same as the TTLS AAA server.

Packet encapsulation

EAP/TTLSv1 packet:  
 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 | Message Length

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

Message Length | Data...

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

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

0 1 2 3 4 5 6 7

+---+---+---+---+---+---+---+---+

| L | M | S | R | R | V |

+---+---+---+---+---+---+---+---+

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.

Security relationships

*Client <-> NAS* - No pre-existing security relationship.

*NAS <-> TTLS AAA Server <-> AAA/H Server* - Pre-existing security relationship is assumed. The secure connection between these points is not in the scope of the EAP/TTLS protocol and has to be configured separately. RADIUS uses pre-shared keys by default to connect with NAS devices (and other RADIUS servers, seen as NAS devices by RADIUS servers).

*Client <-> TTLS AAA Server* - One-way trust based on the servers CA certificate, or two-way trust if client certificate validation is enabled.

Use-case: Client tries to authenticate with a TTLS RADIUS server

1. Client tries to connect to the NAS
2. NAS sends an EAP-Request/Identity message to the client
3. Client responds with an EAP-Response/Identity message (no username / password is present in this message)
4. The NAS now acts as a passthrough device, allowing the TTLS server to negotiate the EAP-TTLS with the client directly

**TLS Handshake:**

1. Server sends an EAP/TTLS start packet to the client
2. Client sends **ClientHello** message with highest supported TLS version and a list of suggested ciphersuites
3. Server sends **ServerHello** message with chosen TLS version and chosen cipersuite (EAP/TTLSv1 here)
4. Server sends certificate to authenticate itself
5. Client verifies this and generates a pre-master secret for the session, ecrypts it with the servers public key (from the server certificate) and (optionally) its own certificate to the server
6. The server (optionally) verifies the client certificate.
7. Using its private key, the server decrypts the client’s pre-master secret.
8. Both the client and the server use the pre-master secret, together with a random number from both Hello messages, to compute the master secret that will be used as the symmetric key to encrypt and decrypt all future communication between server and client.
9. Both server and client send each other a message confirming that future messages will be encrypted. They then send a separate message encrypted with the new master secret, that the client / server part of the TLS handshake is finished  
   **AVP (attribute-value pairs) exchange**
10. Using the master secret, the client now tunnels attribute-value pairs to the EAP/TTLS server, which uses this information to attempt an AAA authentication.
11. … (W.I.P.)

<http://www.painless-security.com/wp/wp-content/uploads/2010/02/moonshot-feasibility-analysis.pdf> (interessante link over Moonshot project (GSSAPI <-> EAP)