Application Note

Configuring PEAP and TTLS in the Interlink Networks RAD-Series and Secure.XS RADIUS Servers

Version 7.0 and higher

This application note explains how to configure the Interlink Networks RAD-Series servers and the Interlink Networks Secure.XS server to do TLS-protected authentication using PEAP[1] or the EAP-TTLS[2] authentication method.

This application note only covers the configuration records in the server configuration files. These are text files and can be edited with a text editor. Use of the Interlink Server Manager for managing server configurations is covered in the Administrator's Guide.

Overview of the TLS-Protected EAP Methods

The EAP-TLS authentication method and the TLS protected EAP methods based on it – EAP-TTLS and PEAP – all make use of the Transport Layer Security (TLS) protocol to provide integrity and confidentiality protection.

The underlying TLS protocol is based on the Secure Sockets Layer (SSL) protocol commonly used by web browsers to secure web transactions. Using public key cryptography, communicating parties may authenticate themselves to each other using public key certificates. In web applications, only the server typically has a certificate and authenticates itself to the client so that the user can have confidence that his communication has not been redirected or intercepted by an imposter. In this case, TLS provides unidirectional authentication. But if both parties have certificates, TLS can provide mutual authentication. Following the authentication phase, the two parties use a key agreement protocol such as Diffie-Hellman to derive a session key which is used to authenticate and encrypt messages exchanged during the TLS session.

EAP-TLS

EAP-TLS uses the TLS public key certificate authentication mechanism within EAP to provide mutual authentication of client to server and server to client. With EAP-TLS, both the client and the server must be assigned a digital certificate signed by a Certificate Authority (CA) that they both trust.

Features of EAP-TLS include:

- Mutual authentication (server to client as well as client to server)
- Key exchange (to establish dynamic WEP or TKIP keys)
- Fragmentation and reassembly (of very long EAP messages, if needed)
- Fast reconnect via TLS session resumption (not currently supported by Interlink)

PEAP

Protected EAP (PEAP) adds a TLS layer on top of EAP in the same way as EAP-TLS, but it then uses the resulting TLS session as a carrier to protect other, legacy EAP methods.

PEAP has an assigned EAP type. Ordinarily PEAP uses TLS only to authenticate the server to the client but not the client to the server. This way, only the server is required to have a public key certificate; the client need not have one. (Although PEAP can theoretically allow the client to use a certificate to authenticate to the server, the Interlink implementation does not allow this. Use EAP-TLS instead.)

After the client is satisfied regarding the authenticity of the server's identity, the client and server exchange a sequence of EAP messages encapsulated within TLS messages. The TLS messages are authenticated and encrypted using TLS session keys negotiated by the client and the server.

The protected EAP messages (those encapsulated within TLS messages) may be of any EAP type desired (except PEAP, TTLS and SPEKE).

PEAP provides the following services to the EAP methods it protects:

- Message authentication (imposters can't insert or falsify EAP messages)
- Message encryption (Imposters can't read or decipher the protected EAP messages)
- Authentication of server to client (only the protected method needs to authenticate client to server)
- Key exchange (to establish dynamic WEP or TKIP keys)
- Fragmentation and reassembly (of very long EAP messages, if needed)
- Fast reconnect via TLS session resumption (not currently supported by Interlink Networks)

PEAP is especially useful as a mechanism to augment the security of legacy EAP methods that lack one or more of the above features. There are many EAP methods that provide adequate security for PPP authentication but completely fail to provide adequate security in a wireless LAN environment. PEAP can therefore be used to augment the security of these legacy methods so that they may adequately be used for 802.1x authentication.[3]

EAP-TTLS

The Tunneled TLS EAP method (EAP-TTLS) is very similar to PEAP in the way that it works and the features that it provides. The difference is that instead of encapsulating EAP messages within TLS, the TLS payload of EAP-TTLS messages consists of a sequence of attributes. By including a RADIUS EAP-Message attribute in the payload, EAP-TTLS can be made to provide the same functionality as PEAP. If, however, a RADIUS Password or CHAP-Password attribute is encapsulated, TTLS can protect the legacy authentication mechanisms of RADIUS. The advantage of this becomes apparent if the TTLS server is used as a proxy to mediate between an access point and a legacy RADIUS home server. When the TTLS server forwards RADIUS messages to the home server, it decapsulates the attributes protected by EAP-TTLS and inserts them directly into the forwarded message. The EAP-TTLS messages are not forwarded to the home server. Thus the legacy authentication mechanisms supported by existing RADIUS severs in the infrastructure can be protected for transmission over wireless LANs.

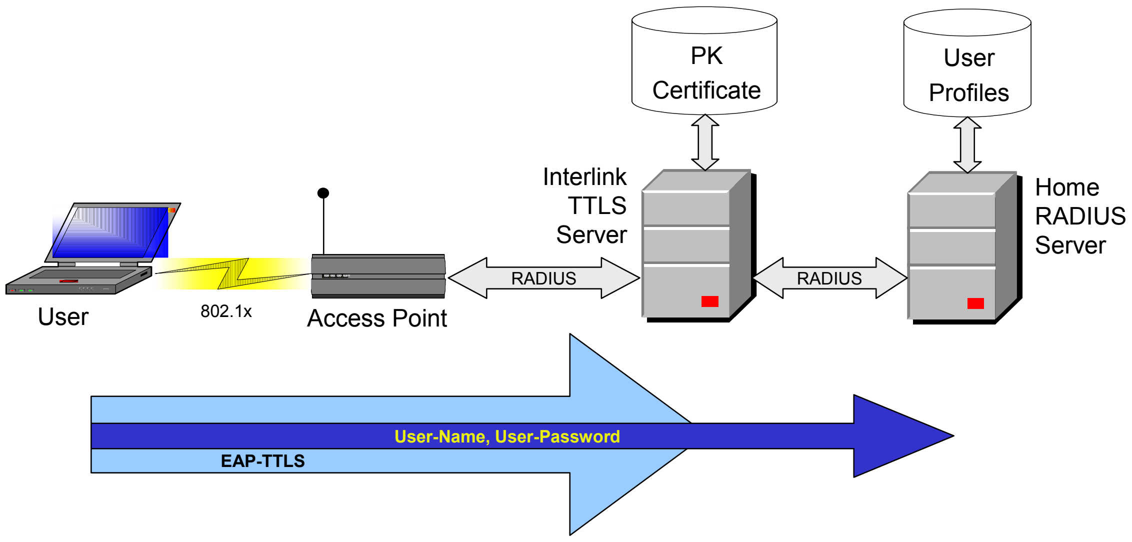

Figure 1 — How a TTLS server interacts with a legacy RADIUS server

Anonymous Identities

Both PEAP and TTLS support identity hiding. In a wireless LAN environment, the access point (AP) typically generates an EAP-Identity request as part of the association process. To preserve anonymity, the EAP client on the user's system may respond with only enough information to allow the first hop RADIUS server to process the request, as shown in the following examples.

Note: Not all PEAP clients support identity hiding.

EAP-Identity = anonymous

In this example, all users will share the pseudo-user-name "anonymous". The first hop RADIUS server is a PEAP or TTLS server which drives the server end of the PEAP or TTLS protocol. The inner (protected) authentication type will then be either handled locally or proxied to a remote (home) RADIUS server.

EAP-Identity = anonymous@realm_x

In this example, users belonging to different realms hide their own identity but indicate which realm they belong to so that the first hop RADIUS server may proxy the PEAP or TTLS requests to RADIUS servers in their home realms which will act as the PEAP or TTLS server. The first hop server acts purely as a RADIUS relay node.

Alternatively, the first hop server may act as the PEAP or TTLS server and either process the protected authentication method or proxy it to another server. This option may be used to configure different policies for different realms.

In PEAP, once the PEAP server and the PEAP client establish the TLS tunnel, the PEAP server generates an EAP-Identity request and transmits it down the TLS tunnel. The client responds to this second EAP-Identity request by sending an EAP-Identity response containing the user's true identity down the encrypted tunnel. This prevents anyone eavesdropping on the 802.11 traffic from discovering the user's true identity.

TTLS works slightly differently. With TTLS, the client typically authenticates via PAP or CHAP protected by the TLS tunnel. In this case, the client will include a User-Name attribute and either a Password or CHAP-Password attribute in the first TLS message sent after the tunnel is established.

With either protocol, the PEAP/TTLS server learns the user's true identity once the TLS tunnel has been established. The true identity may be either in the form user@realm or simply user. If the PEAP/TTLS server is also authenticating the user, it now knows the user's identity and proceeds with the authentication method being protected by the TLS tunnel. Alternatively, the PEAP/TTLS server may forward a new RADIUS request to the user's home server. This new RADIUS request has the PEAP or TTLS protocol stripped out. If the protected authentication method is EAP, the inner EAP messages are transmitted to the home server without the EAP-PEAP or EAP-TTLS wrapper. The User-Name attribute of the outgoing RADIUS message contains the user's true identity – not the anonymous identity from the User-Name attribute of the incoming RADIUS request. If the protected authentication method is PAP or CHAP (supported only by TTLS), the User-Name and other authentication attributes recovered from the TLS payload are placed in the outgoing RADIUS message in place of the anonymous User-Name and TTLS EAP-Message attributes included in the incoming RADIUS request.

Configuring PEAP or TTLS for Realms not Requiring Identity Protection

If you require identity hiding, skip this section and go to the section titled "Configuring PEAP or TTLS for Realms Requiring Identity Protection."

If you do not require identity hiding, configure EAP.authfile entries and authfile entries as described below.

Configuring the EAP.authfile Entries

We will configure the EAP.authfile entries first. You will add a record to this file for each realm for which you will be providing PEAP or EAP-TTLS services. The EAP.authfile records have the following format.

realm EAP *"comment"* { EAP-Type PEAP { *protected-type* } }

or

realm EAP *"comment"* { EAP-TYPE TTLS { *protected-type* } }

where:

- *realm* is replaced by the name of the realm being configured. For those users who do not specify a realm (whose user names are of the form user rather than user@realm, realm\user or realm/user), enter the keyword NULL in place of *realm.*
- EAP specifies EAP authentication is configured for the realm.
- *"comment"* specifies a comment that may be useful to an administrator; if no comment is desired, just specify " ".
- EAP-Type PEAP or EAP-Type TTLS specifies that this server will act as the PEAP or TTLS server for this realm.
- *protected-type* is replaced by the PEAP or TTLS protected authentication type which will be used to authenticate users from this realm. The protected-type field has a syntax all its own which can be somewhat complex. How to specify the *protected-type* is explained below.

How to Enter the Protected-type Field

For either PEAP or TTLS, the *protected-type* field may take either:

- list of EAP-Types

  For realms using TTLS, an additional form is available:

- NULL (no specified *protected-type*)

  Each of these will be explained in detail below.

Protected-type is a List of EAP-Types

If the users for the realm being configured will authenticate with an EAP method protected by PEAP or TTLS, the *protected-type* field should be a list of EAP-Types. An EAP-Type specification has the following format:

```
EAP-Type type
```

where:

- The keyword **EAP-Type** indicates that the protected authentication method is an EAP method.
- The parameter ***type*** is replaced by the name of any EAP method supported by Interlink Networks. A list of these types can be found in the configuration file named dictionary. At the time of this writing, the supported EAP-types are:
  - MD5_Challenge
  - TLS
  - CiscoLEAP
  - TTLS
  - PEAP
  - MS_EAP
  - SPEKE

An example of an EAP-Type specification is:

```
EAP-Type MD5_Challenge
```

The EAP-Type specification can be repeated multiple times, with a different EAP method each time. The EAP-Type specifications are separated by white space.

An example of a list of EAP-Types is:

```
EAP-Type CiscoLEAP  EAP-Type MD5_Challenge
```

In this example, the server will first try to authenticate users from this realm with the CiscoLEAP EAP method. If the user is unable to authenticate with CiscoLEAP, the server will next try the MD5_Challenge EAP method.

When this list is used in an EAP.authfile record, it looks like this:

```
example.net EAP “  ”  {EAP-Type TTLS { EAP-Type CiscoLEAP  EAP-Type MD5_Challenge} }
                      └─────────────────────────v──────────────────────────┘
                                          protected-type
```

Protected-type is Null

TTLS is frequently used to augment the security of legacy authentication systems. Many organizations that use TTLS wish to augment the security of their legacy PAP or CHAP authentication systems for use over wireless LANs. When TTLS is used to protect PAP or CHAP, the TTLS client takes the initiative by encoding the user credentials into RADIUS attributes and presenting them to the TTLS server encapsulated in TLS protected EAP messages. Since the client takes the initiative, no special configuration is needed in the TTLS server.

To configure a realm to do PAP, CHAP, or MS-CHAP authentication protected by TTLS, the protected-type field is null, as shown in the following example:

```
example.net EAP “  ”  { EAP-Type TTLS { } }
                                      └v┘
                                 protected-type
```

or simply:

```
example.net EAP “  ”  { EAP-Type TTLS }
```

Note for Compatibility with the Interlink Server Manager

It is not recommended that you mix hand configuration using a text editor with configuration using the Server Manager, since the Server Manager is not as flexible about line breaks as are the RAD-Series or Secure.XS servers. If you do use both, for each of the constructs above that use braces ({ and }), each left or right brace should be put on a line by itself with only spaces and tabs. Thus the syntax:

realm EAP “ *comment”* { EAP-Type PEAP { *protected-type* } }

should be entered as:

```
realm EAP “ comment”
{
    EAP-Type PEAP
    {
        protected-type
    }
}
```

An example using this expanded syntax is:

```
example.net EAP "  "
{
    EAP-Type PEAP
    {
        EAP-Type CiscoLEAP
        EAP-Type MD5_Challenge
    }
}
```

Configuring the authfile Entries

We have now finished configuring how the users of each realm should be authenticated. The next step is to configure where the user credentials are stored. We shall refer to the database containing the user credentials as the data repository. To configure the data repository for each realm, we turn our attention to the file named **authfile**.

Each realm for which the user authentication will be performed on this server should be configured by adding a record to the file named authfile as follows:

realm repository-type repository

where:

- *realm* is replaced by the name of the realm being configured. For those users who do not specify a realm (whose user names are of the form *user* rather than *user@realm*), enter the keyword NULL in place of *realm*.
- *repository-type* is replaced with the type of repository that contains the user credentials for this realm.
- *repository* is a specification of the data repository from which the user credentials for this realm will be read. The repository types you may configure are:
    - iaaaFile
    - Oracle
    - PROLDAP
    - SAMDatabase

A simple example is:

```
example.net  iaaaFILE  example.net
```

In this example, user credentials for the realm example.net are found in the flat file named example.net.users.

Configuring PEAP or TTLS for Realms Requiring Identity Protection

Note: Not all PEAP clients (the PEAP software that runs on the user's device) support anonymous identities. Before configuring anonymous profiles for PEAP realms, check whether the PEAP clients used in those realms support anonymous identities.

Configuring an Anonymous Profile

The first step is to create configuration profiles for the anonymous identities. The anonymous profiles are configured on the RADIUS server that will act as the PEAP or TTLS server.

You may configure one or more anonymous profiles. Their User-Names may include realms (e.g.: anonymous@realm1, anonymous@realm2) or not

(e.g.: anonymous1, anonymous2).

We recommend that you include the anonymous profile(s) in the users file. That way it will be cached in main memory for efficient reuse. This also allows a User-Name such as anonymous@realm1 to be processed by this server, but the protected user identities in realm1 to be proxied to another server via the authfile if that is desired.

Add the anonymous profile(s) to the configuration file named "**users**" according to the following syntax:

User-Name@Realm Authentication-Type = EAP, EAP-TYPE = PEAP, *check-items*

reply-items

or:

User-Name@Realm Authentication-Type = EAP, EAP-TYPE = TTLS, *check-items*

reply-items

where:

- *User-Name@Realm* is replaced by the identifier for the anonymous profile. The @Realm portion is optional and may be omitted for users that do not specify a realm.
- Authentication-Type = EAP specifies that users that originally identify themselves as anonymous should be authenticated using EAP.
- EAP-TYPE = PEAP or EAP-TYPE = TTLS further specifies that EAP-PEAP or EAP-TTLS, respectively, should be used to authenticate users claiming this anonymous identity.
- *check-items* may be optionally replaced with a list of check and deny items that will apply to all users who begin authentication by claiming this anonymous identity. These check items will be checked after the user's true identity is authenticated and after any policy, check/deny,

or reply items associated with the user's true profile have been applied. The *check-items* must be included on the same line as the identifier.

- *reply-items* may be optionally replaced with a list of attributes which will be added to the user's Access Request and (assuming successful authentication) Access Accept messages. Unlike the check/deny items, the reply items are added to the user attribute list before user authentication. They may be overridden by subsequent reply items from the user's true profile. The reply-items must be placed on lines following the identifier and beginning with one or more tabs or spaces.

Configuring the User Authentication to be Done on Another RADIUS Server

You have two options open to you as to where the actual user authentication will take place.

One option is to perform the actual user authentication directly on the PEAP/TTLS server. To implement this option, skip this section and proceed directly to the section titled "Configuring the User Profiles".

The other option is for the PEAP/TTLS server to forward RADIUS messages containing the protected authentication mechanism to another RADIUS server for authentication and authorization. The forwarded requests will not contain any EAP-PEAP or EAP-TTLS messages. Rather, they will contain the EAP messages that were protected (encapsulated) by the EAP-PEAP or EAP-TTLS messages or, in the case of TTLS, the PAP-Password or CHAP-Password and CHAP-Challenge attributes that were protected (encapsulated) by the EAP-TTLS messages. This section describes how to configure this option.

All requests for which the User-Name cannot be found in the **users** file will be authenticated as specified in the configuration file named "**authfile**". This will cause PEAP and TTLS requests to be authenticated as specified in the authfile once the true User-Name is discovered.

Each realm for which the user authentication will be done on another RADIUS server should be configured by adding a record to the file named authfile as follows:

realm RADIUS *server*

where:

- *realm* is replaced by the name of the realm being configured. For those users who do not specify a realm (whose user names are of the form *user* rather than *user@realm*), enter the keyword NULL in place of *realm.*
- RADIUS specifies that all requests for this realm other than anonymous requests should be proxied via RADIUS to the specified server.
- *server* is replaced by the name of the RADIUS server to which requests for this realm should be proxied.

Configuring the User Profiles

The profiles for the individual users who will authenticate using a TLS-protected authentication method (one protected by EAP-PEAP or EAP-TTLS) are configured in exactly the same way they would be configured if PEAP or TTLS protection were not being used.

If EAP-TTLS is the protecting protocol, then individual users may be authenticated via PAP, CHAP, MS-CHAP, or EAP. If PEAP is the protecting protocol, then individual users must be authenticated by some EAP method other than PEAP, TTLS or SPEKE.

Limitations of the Interlink PEAP and TTLS implementations

As of version 6.1 of the RAD-Series servers or Secure.XS server, the following restrictions apply:

- TLS session resumption is not supported for the EAP-TLS, EAP-TTLS, or EAP-PEAP EAP methods.
- Support for RADIUS accounting when EAP-PEAP or EAP-TTLS is used is limited as follows. RADIUS accounting records may be logged on the PEAP/TTLS server, but if the user employs identity hiding, they will all bear the anonymous user name – not the protected individual user name. If individual authentication is done on a RADIUS server other than the PEAP/TTLS server and identity protection is employed, then the home server will not receive RADIUS accounting records.
- LAS session records may be maintained on the PEAP/TTLS server if the access point is configured to generate RADIUS accounting messages. The user name field of these session records will be the anonymous user name initially specified by the user, however, and not the individual's true user name if identity protection is employed. If the individual authentication is done on a RADIUS server other than the PEAP/TTLS server and identity protection is employed, then LAS session records may not be maintained on the home server.
- Simultaneous use restrictions may be configured for the maximum number of times an anonymous user name may simultaneously be in use, but simultaneous use restrictions on individual users sharing an anonymous initial identity may not be configured.
- The PEAP and TTLS implementations do not support mutual authentication in part (or phase) 1 and therefore do not permit the omission of a protected authentication mechanism.
- Unfortunately, it is not possible to enforce that users must authenticate using PEAP or TTLS protection. If they attempt to authenticate using an allowable authentication method directly, the authentication will succeed.

References

1 Anderson, Hakan, et al, “Protected EAP Protocol (PEAP),” Internet Draft draft-josefsson-pppext-eap-tls-eap-05.txt, Sept. 2002, work in progress.

2 Funk, Paul and Simon Blake-Wilson, “EAP Tunneled TLS Authentication Protocol (EAP-TTLS)”, Internet Draft draft-ietf-pppext-eap-ttls-02.txt, Nov. 2002, work in progress.

3 “IEEE Standard for Local and metropolitan area networks – Port-Based Network Access Control”, IEEE Std 802.1x-2001, June 2001.

4 “Using 802.1X for Wireless Local Area Networks with Interlink Networks Software”, Interlink Networks, Inc., 2002.