Como hacer una red wifi segura:
WPA + EAP-TLS + RADIUS

Toni de la Fuente Díaz
http://blyx.com

6 Julio 2005

Objetivos:

Vamos a configurar una red wifi segura usando certificados de cliente y servidor para la autenticación. Para montar la red netcesitamos:

-Tener ganas (importante).
-Un servidor RADIUS.
-Un AP con soporte WPA y EAP (802.1X)
-Un cliente wifi.

Como servidor RADIUS vamos a utilizar Linux Fedora Core 2 con el sistema de instalación de paquetes apt y como software RADIUS usaremos FreeRADIUS.

El AP (Access Point) será un Dlink 2000AP+ (802.11b/g) con la última versión de firmware instalada para tener soporte WPA en el dispositivo. Hay que tener en cuenta que este hardware se fabricó sin tener en cuenta WPA así que se ha añadido dicha funcionalidad en el firmware por lo que no soportará muchos clientes.

El cliente usado será un portátil con MAC OS X (10.4) cuyo soporte WPA está incluido de forma nativa. No obstante se indican sitios de referencia para configurar otros sistemas operativos.

Preparando el sistema, instalación de apt para Fedora:

Instalación de apt para Fedora Core 2
wget http://ftp.freshrpms.net/pub/freshrpms/fedora/linux/2/apt/apt-0.5.15cnc6-1.1.fc2.fr.i386.rpm

rpm -hiv apt-0.5.15cnc6-1.1.fc2.fr.i386.rpm

Ampliamos la lista de repositorios
vi /etc/apt/sources.list.d/dag.list
rpm http://apt.sw.be fedora/2/en/i386 dag

apt-get update

Instalación y configuración de FreeRADIUS, el cliente RADIUS y utilidades OpenSSL:

apt-get install freeradius radiusclient openssl-perl

Hacemos que FreeRADIUS se inicie cada vez que arranca el sistema operativo:

chkconfig radiusd on

Todos los archivos de configuración de FreeRADIUS se encuentran en:
/etc/raddb/

Cabe destacar los siguientes archivos de configuración:
radiusd.conf - Archivo general de configuración de FreeRADIUS y del daemon.

eap.conf – Archivo de configuración de las directivas EAP a utilizar. Es un include de radiusd.conf

clients.conf – Descripción y credenciales de los diferentes dispositivos que consultan al RADIUS (Aps, NAS, etc).

users – Archivo donde se especifican las credenciales de los usuarios de la red. Se usa este archivo si no existe otro backend para el almacenamiento de los usuarios.

He subido a mi sitio web un tar.gz con todos los archivos de configuracóin, certificados y scripts para crear los certificados (de CA, de servidor y de clientes), puedes descargar el paquete de la siguiente forma:

wget http://blyx.com/public/wireless/wpa+eap-tls+radius/raddb.tar.gz

Copiamos los archivos descargados a su ubicación original, es decir, /etc/raddb.

Al estar basándonos en la arquitectura PKI, necesitamos generar un conjunto de certificados basados en el modelo cliente/servidor que funcionarán como armazón del proceso de autenticación. Esto significa que debemos crear una Autoridad de Certificación (CA) y generar los certificados tanto para el servidor como para cada cliente.

Vamos a crear los certificados:

Para crear la CA y los certificados se pueden usar los scripts que aparecen en la web http://www.alphacore.net/contrib/nantes-wireless/eap-tls-HOWTO.html. No obstante, como he dicho antes, están disponibles en el raddb.tar.gz que hemos descargado previamente. Aquí los vemos con algunas modificaciones que permiten el correcto funcionamiento en Fedora Core 2.

cd /etc/raddb/certs

vi xpextensions

[xpclient_ext]
extendedKeyUsage = 1.3.6.1.5.5.7.3.2
[xpserver_ext]
extendedKeyUsage = 1.3.6.1.5.5.7.3.1

vi CA.root

```
#!/bin/sh
# needed if you need to start from scratch otherwise the CA.pl -newca command doesn't copy the new
# private key into the CA directories
rm -rf demoCA
echo "*****************************************************************************"
echo "Creating self-signed private key and certificate"
echo "When prompted override the default value for the Common Name field"
echo "*****************************************************************************"
echo
# Generate a new self-signed certificate.
# After invocation, newreq.pem will contain a private key and certificate
# newreq.pem will be used in the next step
openssl req -new -x509 -keyout newreq.pem -out newreq.pem -passin pass:whatever -passout
pass:whatever
echo "*****************************************************************************"
echo "Creating a new CA hierarchy (used later by the "ca" command) with the certificate"
echo "and private key created in the last step"
echo "*****************************************************************************"
echo
echo "newreq.pem" | /usr/share/ssl/misc/CA.pl -newca >/dev/null
echo "*****************************************************************************"
echo "Creating ROOT CA"
echo "*****************************************************************************"
echo
# Create a PKCS#12 file, using the previously created CA certificate/key
# The certificate in demoCA/cacert.pem is the same as in newreq.pem. Instead of
# using "-in demoCA/cacert.pem" we could have used "-in newreq.pem" and then omitted
# the "-inkey newreq.pem" because newreq.pem contains both the private key and certificate
openssl pkcs12 -export -in demoCA/cacert.pem -inkey newreq.pem -out root.p12 -cacerts -passin
pass:whatever -passout pass:whatever
# parse the PKCS#12 file just created and produce a PEM format certificate and key in root.pem
openssl pkcs12 -in root.p12 -out root.pem -passin pass:whatever -passout pass:whatever
# Convert root certificate from PEM format to DER format
openssl x509 -inform PEM -outform DER -in root.pem -out root.der
#Clean Up
rm -rf newreq.pem
```

vi CA.server

```
#!/bin/sh
echo "*****************************************************************************"
echo "Creating server private key and certificate"
echo "When prompted enter the server name in the Common Name field."
echo "*****************************************************************************"
echo
# Request a new PKCS#10 certificate.
# First, newreq.pem will be overwritten with the new certificate request
openssl req -new -keyout newreq.pem -out newreq.pem -passin pass:whatever -passout pass:whatever
# Sign the certificate request. The policy is defined in the openssl.cnf file.
# The request generated in the previous step is specified with the -infiles option and
# the output is in newcert.pem
# The -extensions option is necessary to add the OID for the extended key for server authentication
openssl ca -policy policy_anything -out newcert.pem -passin pass:whatever -key whatever -extensions
xpserver_ext -extfile xpextensions -infiles newreq.pem
# Create a PKCS#12 file from the new certificate and its private key found in newreq.pem
# and place in file specified on the command line
openssl pkcs12 -export -in newcert.pem -inkey newreq.pem -out $1.p12 -clcerts -passin pass:whatever
-passout pass:whatever
# parse the PKCS#12 file just created and produce a PEM format certificate and key in certsrv.pem
openssl pkcs12 -in $1.p12 -out $1.pem -passin pass:whatever -passout pass:whatever
# Convert certificate from PEM format to DER format
openssl x509 -inform PEM -outform DER -in $1.pem -out $1.der
# Clean Up
rm -rf newert.pem newreq.pem
```

cat CA.client

```
#!/bin/sh
echo "*****************************************************************************"
echo "Creating client private key and certificate"
echo "When prompted enter the client name in the Common Name field. This is the same"
echo " used as the Username in FreeRADIUS"
echo "*****************************************************************************"
echo
# Request a new PKCS#10 certificate.
# First, newreq.pem will be overwritten with the new certificate request
```

```
openssl req -new -keyout newreq.pem -out newreq.pem -passin pass:whatever -passout pass:whatever
# Sign the certificate request. The policy is defined in the openssl.cnf file.
# The request generated in the previous step is specified with the -infiles option and
# the output is in newcert.pem
# The -extensions option is necessary to add the OID for the extended key for client authentication
openssl ca -policy policy_anything -out newcert.pem -passin pass:whatever -key whatever -extensions
xpclient_ext -extfile xpextensions -infiles newreq.pem
# Create a PKCS#12 file from the new certificate and its private key found in newreq.pem
# and place in file specified on the command line
openssl pkcs12 -export -in newcert.pem -inkey newreq.pem -out $1.p12 -clcerts -passin pass:whatever
-passout pass:whatever
# parse the PKCS#12 file just created and produce a PEM format certificate and key in certclt.pem
openssl pkcs12 -in $1.p12 -out $1.pem -passin pass:whatever -passout pass:whatever
# Convert certificate from PEM format to DER format
openssl x509 -inform PEM -outform DER -in $1.pem -out $1.der
# clean up
rm -rf newcert newreq.pem
```

chmod 700 CA.*

./CA.root

```
********************************************************************************
Creating self-signed private key and certificate
When prompted override the default value for the Common Name field
********************************************************************************

Generating a 1024 bit RSA private key
..............................++++++
.....................................++++++
writing new private key to 'newreq.pem'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [GB]:ES
State or Province Name (full name) [Berkshire]:Granada
Locality Name (eg, city) [Newbury]:Granada
Organization Name (eg, company) [My Company Ltd]:Blyx
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []:
Email Address []:
********************************************************************************
Creating a new CA hierarchy (used later by the ca command) with the certificate
and private key created in the last step
********************************************************************************

********************************************************************************
Creating ROOT CA
********************************************************************************

MAC verified OK
```

uname -n
radius.blyx.com

./CA.server radius.blyx.com

```
********************************************************************************
Creating server private key and certificate
When prompted enter the server name in the Common Name field.
********************************************************************************

Generating a 1024 bit RSA private key
.............++++++
...++++++
writing new private key to 'newreq.pem'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [GB]:ES
State or Province Name (full name) [Berkshire]:Granada
Locality Name (eg, city) [Newbury]:Granada
Organization Name (eg, company) [My Company Ltd]:Blyx
```

```
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []:radius.blyx.com
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
Using configuration from /usr/share/ssl/openssl.cnf
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 1 (0x1)
        Validity
            Not Before: Jul  7 17:01:54 2005 GMT
            Not After : Jul  7 17:01:54 2006 GMT
        Subject:
            countryName               = ES
            stateOrProvinceName       = Granada
            localityName              = Granada
            organizationName          = Blyx
            commonName                = radius.blyx.com
        X509v3 extensions:
            X509v3 Extended Key Usage:
            TLS Web Server Authentication
Certificate is to be certified until Jul  7 17:01:54 2006 GMT (365 days)
Sign the certificate? [y/n]:y


1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
MAC verified OK
```

Este comando debería ejecutarse por cada usuario:

```
# ./CA.client toni
********************************************************************************
Creating client private key and certificate
When prompted enter the client name in the Common Name field. This is the same
 used as the Username in FreeRADIUS
********************************************************************************

Generating a 1024 bit RSA private key
..............++++++
.++++++
writing new private key to 'newreq.pem'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [GB]:ES
State or Province Name (full name) [Berkshire]:Granada
Locality Name (eg, city) [Newbury]:Granada
Organization Name (eg, company) [My Company Ltd]:Blyx
Organizational Unit Name (eg, section) []:Blyx
Common Name (eg, your name or your server's hostname) []:toni
Email Address []:toni@blyx.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
Using configuration from /usr/share/ssl/openssl.cnf
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 2 (0x2)
        Validity
            Not Before: Jul  7 17:04:37 2005 GMT
            Not After : Jul  7 17:04:37 2006 GMT
        Subject:
            countryName               = ES
            stateOrProvinceName       = Granada
            localityName              = Granada
            organizationName          = Blyx
            organizationalUnitName    = Blyx
            commonName                = toni
            emailAddress              = toni@blyx.com
        X509v3 extensions:
            X509v3 Extended Key Usage:
            TLS Web Client Authentication
```

```
Certificate is to be certified until Jul  7 17:04:37 2006 GMT (365 days)
Sign the certificate? [y/n]:y


1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
MAC verified OK
```

Despues de generar los certificados deberemos copiar los archivos root.der y toni.p12 (o el <usuario_que_sea>.p12) al ordenador del cliente e instalarlos, esto lo veremos más adelante.

Para configurar el servidor RADIUS utilizaremos root.pem y radius.blyx.com.pem (o el <nombre_del_servidor>.pem)

Ahora vamos a probar que la configuración del servidor RADIUS la hemos hecho de forma correcta:

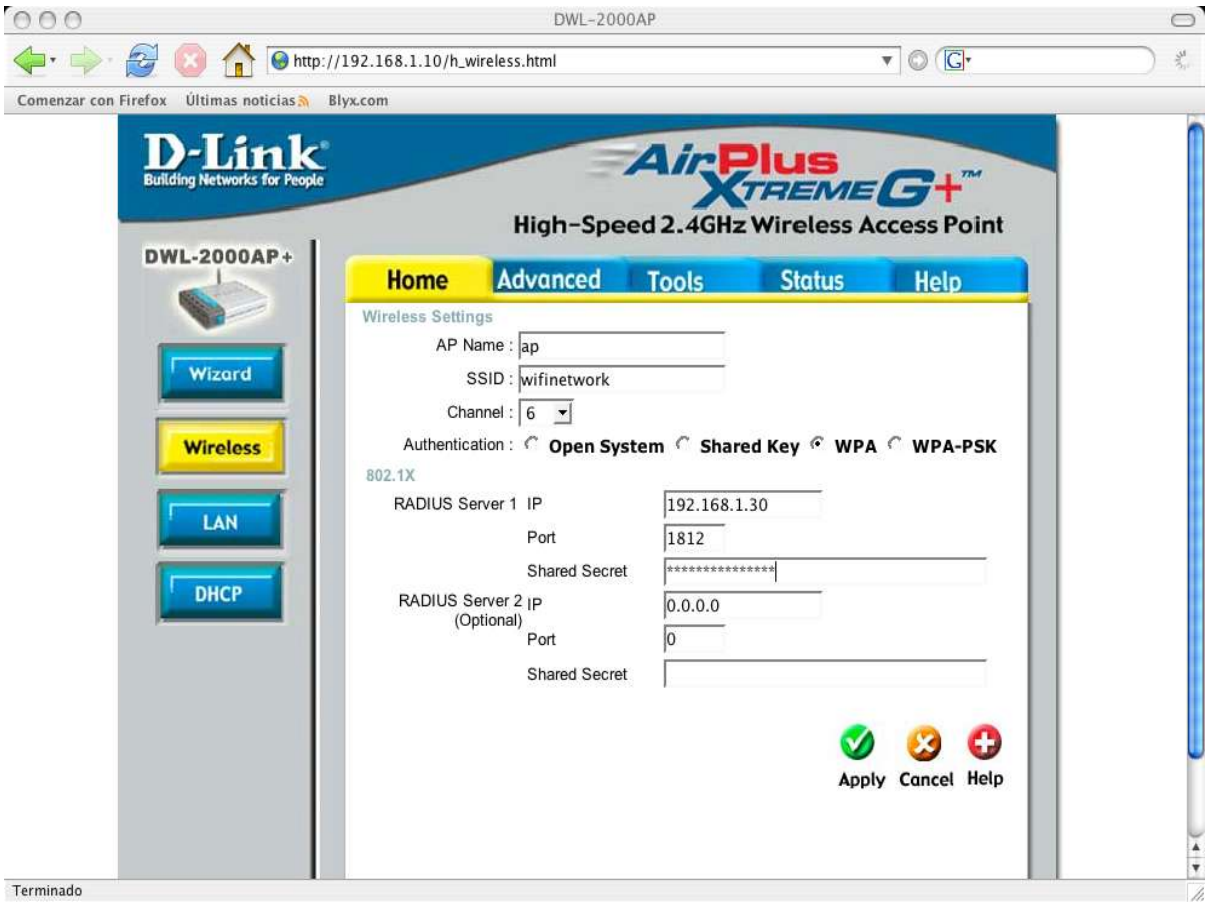
radiusd -X
Ready to process requests.

Si todo ha ido bien, pulsamos Ctl+C y arrancamos el servicio:
/etc/init.d/radiusd start

Podemos ver los logs del servidor de la siguiente forma:
tail -f /var/log/radius/radius.log

Configuración del AP:

Activamos la configuración en el AP. Seleccionamos WPA, indicamos la IP del servidor RADIUS (recuerda hay que abrir los puertos 1812/UDP y 1813/UDP en el firewall local del servidor RADIUS). **SharedSecret** es el valor que le hemos dado al atributo **secret** que se encuentra en el archivo **clients.conf**, en nuestro caso usaremos **laboratorio2005**. Esta clave es usada para cifrar la comunicación entre el cliente RADIUS (AP) y el servidor RADIUS. Como ves, es posible añadir otro servidor RADIUS para usarlo como failover, es decir, en caso de no estar disponible el primero usaría el segundo.



El cliente:

No todas las tarjetas wifi soportan WPA (depende del firmware y/o hardware) y 802.1X así que puedes comprobar la tuya en la siguiente dirección: http://wireless.utah.edu/cgi-bin/dot1x/dot1xCompatibility.pl

En este caso vamos a ver como configurar un cliente Mac OS X 10.4 sobre un PowerBook G4, ya que la tarjeta wifi interna de este portátil soporta WPA de forma nativa así como el SO versión superior o igual a 10.3.8.

La configuración de clientes para acceso a redes 802.1X – EAP-TLS es un poco mas “entretenida” que otras variantes de EAP y es soportada por Linux, Mac OS X, FreeBSD, Solaris y Windows XP SP2, unos de forma nativa y otros con un cliente espeífico:

Mac OS X:

-Soporte nativo del sistema.
-AEGIS Client http://www.mtghouse.com

Linux:

-Xsupplicant: http://www.open1x.org/
-AEGIS Client http://www.mtghouse.com
-wpa_supplicant http://hostap.epitest.fi/wpa_supplicant

FreeBSD:

-PANA: http://www.opendiameter.org/

Windows:

-Soporte nativo del sistema Windows XP SP2.
-WIRE1x: http://wire.cs.nthu.edu.tw/wire1x/
-AEGIS Client (98/CE/Me/2K/NT4) http://www.mtghouse.com

Solaris:

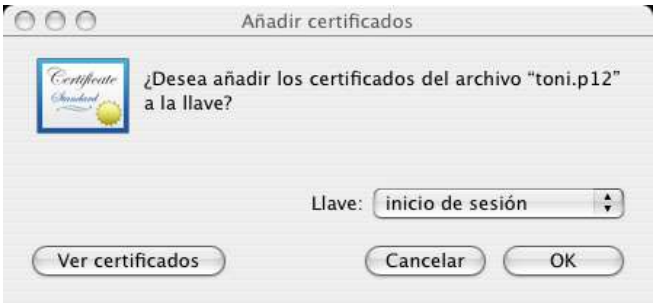
-AEGIS Client http://www.mtghouse.com

Configuración del cliente Mac OS X 10.4:

Vamos a configurar el cliente en nuestro flamante Tiger.

Lo primero que debemos hacer es copiar a nuestra máquina cliente los ficheros creados anteriormente en el servidor RADIUS, estos ficheros son: toni.p12 y root.der. La forma más segura es a través de SSH.

Ahora vamos a instalar el certificado de usuario. Hacemos doble click en el archivo toni.p12 y se nos abrirá la siguiente ventana que forma parte de la aplicación "Acceso a llaves":



Aceptamos el certificado "OK" y a continuación introducimos la clave con la que hemos generado el certificado, si analizamos los scripts de creación podemos ver que esta clave es "whatever" (la podemos cambiar, por supuesto). Pinchamos en "OK".

Ahora debemos crear un certificado para unificar la autenticación basada en certificados. Para ello pinchamos en Archivo -> Nueva llave...

Asignamos contraseña a la llave:

Arrastra la Clave privada que está en “Inicio de sesión” -> “Claves” a la llave que hemos creado “Llave de toni”

Ahora vamos a configurar la conexión 802.1X (EAP-TLS)

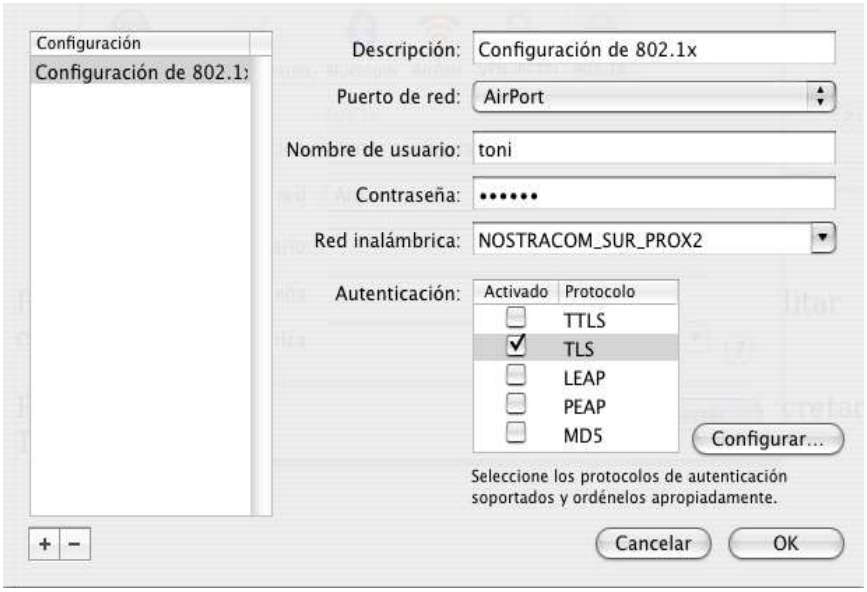
Aplicaciones -> Conexión a Internet

Pincha en Archivo -> Nueva conexión 802.1X...

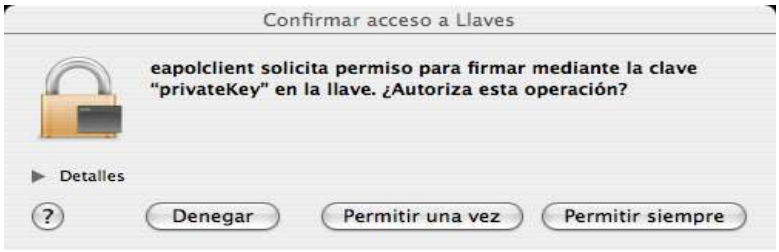
Ahora podemos ver en la ventada “Conexión a Internet” un nuevo item llamado “802.1X”:

Pinchamos en “Configuración de 802.1x” y luego en “Editar configuraciones...”

Podemos ver que Tiger soporta varios tipos de EAP, concretamente, TTLS, TLS, LEAP, PEAP y MD5. Nosotros usaremos TLS.



Indicamos el nombre de usuario para el cual hemos generado el certificado, la contraseña con la que ciframos el certificado “whatever”, seleccionamos el punto de acceso al que queremos acceder, en mi caso “NOSTRACOM_SUR_PROX2”, seleccionamos el método de autenticación y activamos TLS. Para seleccionar el certificado que vamos a usar pinchamos en “Configurar…” y seleccionamos nuestro certificado, en mi caso “toni”. Aceptamos pinchando en “OK” y luego “Conectar”.



Pinchamos en “Permitir siempre”

Nos configuramos una IP, si no tenemos DHCP configurado y listo ;-)

Podemos ver en el RADIUS el siguiente log:
tail -f /var/log/radius/radius.log

```
Thu Jul  7 20:39:52 2005 : Info: rlm_eap_tls:  Received EAP-TLS First Fragment of the message
Thu Jul  7 20:39:52 2005 : Info:     (other): SSL negotiation finished successfully
Thu Jul  7 20:39:52 2005 : Info: rlm_eap_tls: Received EAP-TLS ACK message
Thu Jul  7 20:39:52 2005 : Auth: Login OK: [toni/<no User-Password attribute>] (from client
localhost port 0 cli 00-11-24-90-67-a1)
```

Referencias:

Wi-Foo: The secrets of wireless hacking. Andrew A. Vladimirov, Konstantin V. Gavrilenko, Andrei A. Mikhailovsky. http://www.wi-foo.com
http://www.freeradius.org/doc/EAPTLS.pdf
http://www.missl.cs.umd.edu/wireless/eaptls/?tag=missl-802-1
http://www.alphacore.net/contrib/nantes-wireless/eap-tls-HOWTO.html

http://www.fi.infn.it/system/WiFi/802.1X/macosx/
http://www.dartmouth.edu/~pkilab/greenpass/gp-web-images/internetconnect2.png
http://www.alphacore.net/spipen/article.php3?id_article=1
http://oriol.joor.net/blog-dev/?itemid=1574