Apéndice D - Instalación y configuración de servidor RADIUS

D1. Sistema operativo

El servidor RADIUS se escogió por ser robusto en su seguridad, el paquete Freeradius es *software* libre además de estable y fácil de utilizar. La elección del sistema operativo fue un problema, se probó con las distribuciones Redhat, Debían y Ubuntu dando inconvenientes al crear el mensaje EAP e incompatibilidades con la información del cliente, la distribución que presentó estabilidad para el servicio de autenticación fue Opensuse en la versión 10.2. Actualmente el servidor RADIUS soporta los subtipos de la Tabla D1 del protocolo de autenticación extensible (EAP, *Extensible Authentication Protocol*).

Subtipo EAP	Elección
Cisco LEAP	No se usó porque el cliente debe soportarlo y no se utiliza para Windows
EAP MD5	No se uso porque no es muy soportando en clientes móviles
EAP MSCHAPv2 (PEAPv0)	Soportado en Windows y muchas aplicaciones propietarias
EAP GTC (PEAPv1)	Soportado por muchas aplicaciones propietarias y no se uso en Windows porque es necesario modulo SecureW2
EAP SIM	Usa un dispositivo GSM para la autenticación y distribución de claves
EAP TLS	Soportado por Windows y muchas aplicaciones propietarias
EAP TTLS (PAP-CHAP-MSCHAP)	Soportado por muchas aplicaciones propietarias y no se uso en Windows porque es necesario modulo SecureW2
Para más información o descarga del modulo SecureW2 ir a http://www.securew2.org	

Tabla D1 Subtipos EAP soportados por servidor RADIUS.

La instalación se hizo descargando el *software* [1] de la distribución Linux se actualiza el catalogo de *software* YAST con los repositorios se pueden ver en la Figura D1, oss, non-oss. Update, src-non-oss,src-oss.

Estado	Regenerar	Nombre	URL
en	en	YaST	http://download.opensuse.org/distribution/10.2/repo/non-oss/
en	en	YUM	http://ftp5.gwdg.de/pub/suse/update/10.2/
en	en	YUM	http://download.opensuse.org/distribution/10.2/repo/src-oss/suse/
en	en	YaST	http://download.opensuse.org/distribution/10.2/repo/oss/
en	Apagado	YaST	cd:///?devices=/dev/hda

Figura D1 Diferencias del proceso de autenticación.

D2. Instalación de Openssl y Freeradius

Por medio de YAST se instala todo lo relacionado con Openssl [2] y Freeradius [3], en la Figura D2 se pueden ver los paquetes necesarios para el buen funcionamiento de la autenticación y certificación por medio del servidor RADIUS.

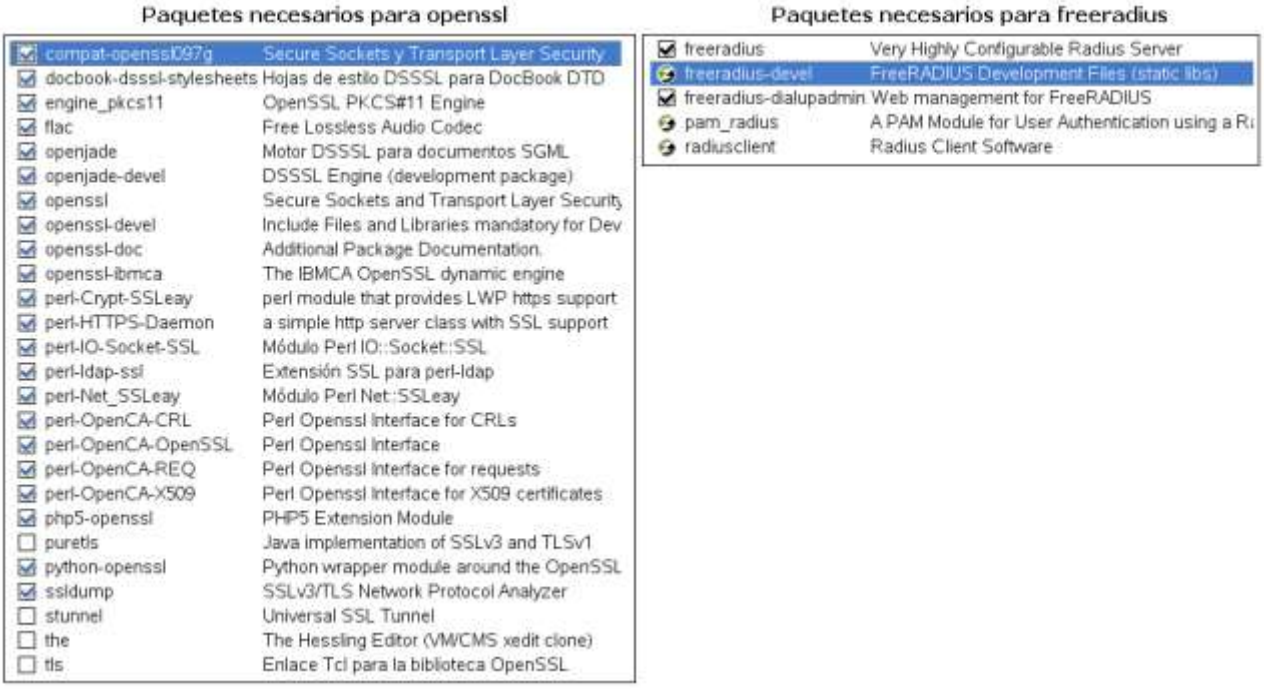

Figura D2 Paquetes de instalación para Freeradius.

D3. Configuración de Freeradius

Los archivos encontrados en la ruta `/etc/raddb` incluyen las instrucciones necesarias para poder ejecutar el servidor de autenticación, lo principal es la creación de certificados personalizados por cada cliente a autenticar, en la ruta `/etc/raddb/cert/` se genera el archivo `xpextensions` con la siguiente información.

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

La creación de certificados es un proceso que exige el manejo de 3 archivos con contenido que junto al paquete de SSL forman automáticamente los certificados pidiendo información para el perfecto control de la conexión, estos son: `CA.root`, `CA.server`, `CA.client`, los certificados digitales deben ser ubicados en cada computador o terminal móvil y ser instalados.

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
```

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
```

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
```

Los 3 archivos deben tener permisos totales para lectura, escritura y ejecución, se ejecuta el primero `#./CA.root` , los primeros 4 campos de datos son referentes a localización y nombre de organización, la información debe ser contenida en los 3 archivos que para este caso son: `CO` (iníciales de Colombia), `Cauca, Popayán` y `Unicauca` siguiendo el orden.

El archivo `CA.server` se debe ejecutar junto con el nombre del servidor `#./CA.server dhcppc0`, se ingresa la información referente a nombre común en este caso dhcppc0 y los campos de contraseña se dejan en blanco para no tener problemas en caso de olvido, el archivo `CA.client` se debe ejecutar con el nombre del usuario y la clave por defecto `whatever` de la siguiente manera `#./CA.client Andres whatever`, después de generados los certificados se copian e instalan los archivos `root.der` y `Andres.p12` o `usuario.p12` en el equipo del cliente.

Adicional a la creación de certificados se deben configurar los archivos del servicio de RADIUS como `users, radiusd.conf, clients.conf, eap.conf;` los archivos `client.conf` y `radiusd.conf` se configuran y se mantienen independientes del protocolo de autenticación seleccionado, los demás, cambian en algunas partes dependiendo del bloque de seguridad que se vaya a utilizar.

En el archivo `clients.conf` se configuran los datos de los puntos de acceso que accederán al servidor RADIUS.

```
client 192.168.1.1 {
       secret              = perseo
       shortname           = DD-WRT1
       nastype             = other
                     }
```

En el archivo `radiusd.conf` se deben cambiar y habilitar las siguientes líneas:

```
mschap {
              authtype = MS-CHAP
              use_mppe = yes
              require_encryption = yes
         }

authorize {
              chap
              mschap
              eap
            }
authenticate {

       Auth-Type PAP {
              pap
       }
       Auth-Type CHAP {
              chap
       }
       Auth-Type MS-CHAP {
              mschap
       }
       eap
       }
```

Bloque de autenticación PEAP con posibilidad de certificación y validación de nombre de servidor. En el archivo `users` se crean los usuarios con sus respectivas contraseñas:

Para MSCHAP-V2.

```
"Andres Arce" Auth-Type := EAP, User-Password == "Andres Arce"
```

Para GTC.

```
"Andres Arce"       User-Password == "Andres Arce"
```

En el archivo `eap.conf` se configura el bloque del protocolo PEAP.

```
eap {
                     default_eap_type = peap

       gtc {
                     auth_type = local
           }
       tls {
                     private_key_password = whatever
                     private_key_file = ${raddbdir}/certs/dhcppc0.pem
                     certificate_file = ${raddbdir}/certs/dhcppc0.pem
                     CA_file = ${raddbdir}/certs/root.pem
                     dh_file = ${raddbdir}/certs/dh
                     random_file = ${raddbdir}/certs/random
                     fragment_size = 1024
                     include_length = yes
                     check_cert_cn = %{User-Name}
                     cIPher_list = "DEFAULT"
              }
}
```

Para MSCHAP-V2.

```
peap {
                default_eap_type = mschapv2
        }
```

Para GTC.

```
peap {
                default_eap_type = gtc
        }
```

Bloque de autenticación TTLS con posibilidad de certificación y validación de nombre de servidor. En el archivo `users` se crean los usuarios con sus respectivas contraseñas:

```
"Andres Arce"       User-Password == "Andres Arce"
```

En el archivo `eap.conf` se configura el bloque del protocolo TTLS.

```
eap {               default_eap_type = ttls

      tls {
                private_key_password = whatever
                private_key_file = ${raddbdir}/certs/dhcppc0.pem
                certificate_file = ${raddbdir}/certs/dhcppc0.pem
                CA_file = ${raddbdir}/certs/root.pem
                dh_file = ${raddbdir}/certs/dh
                random_file = ${raddbdir}/certs/random
                fragment_size = 1024
                include_length = yes
                check_cert_cn = %{User-Name}
                cIPher_list = "DEFAULT"
        }

      ttls {
                default_eap_type = md5
        }
}
```

Bloque de autenticación TLS con posibilidad de certificación y validación de nombre de servidor. En el archivo `users` se crean los usuarios con sus respectivas contraseñas:

```
"Andres Arce"       User-Password == "Andres Arce"
```

En el archivo `eap.conf` se configura el bloque del protocolo TTLS.

```
eap {               default_eap_type = tls
      tls {
                private_key_password = whatever
                private_key_file = ${raddbdir}/certs/dhcppc0.pem
                certificate_file = ${raddbdir}/certs/dhcppc0.pem
                CA_file = ${raddbdir}/certs/root.pem
                dh_file = ${raddbdir}/certs/dh
                random_file = ${raddbdir}/certs/random
                fragment_size = 1024
                include_length = yes
                check_cert_cn = %{User-Name}
                cIPher_list = "DEFAULT"
        } }
```

Se inicia el servidor RADIUS con la orden `radiusd -x` en una terminal de comandos y de esta forma la petición será aceptada perfectamente y para detener el servicio se escribe en consola `/etc/init.d/radiusd stop`.

Los equipos inalámbricos seleccionados para la prueba fueron D-link, los cuales se les actualizó el *Filmware* a DD-WRTv24 RC4 (Para D-Link - DIR 300) que brinda más posibilidades de configuración. Ya configurado el servidor de autenticación hay que complementar la capa de seguridad con la selección de tipo de autenticación empresa (Actualización de punto de acceso en el Apéndice H).

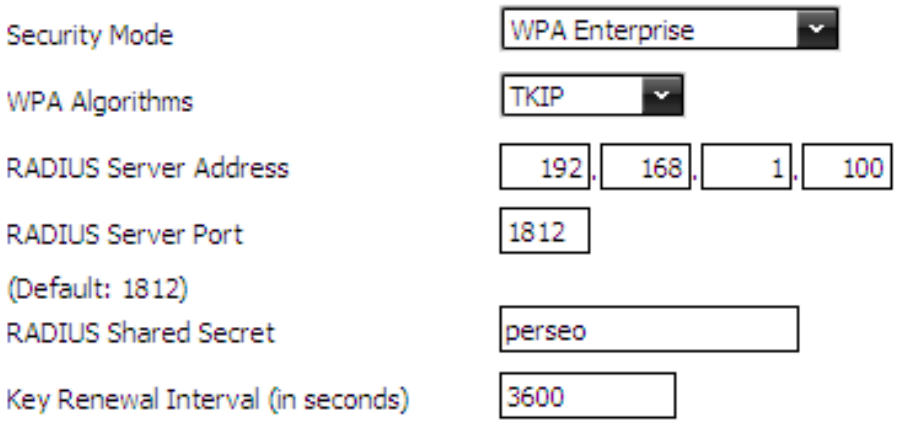

Figura D3 Campos de configuración para el AP.

Son importantes la dirección del servidor RADIUS, si todo el direccionamiento lo mantiene un servidor DHCP entonces verificar cada vez que las direcciones correspondan, el puerto de servidor RADIUS es por defecto el 1812 y la contraseña secreta compartida entre el servidor y el AP.

REFERENCIAS BIBLIOGRÁFICAS

[1] OPENSUSE, Novel, Disponible en: <http://software.opensuse.org/old/10.2>
[2] OPENSSL, Holger Reif, Paul C. Sutton, Disponible en: <http://www.openssl.org>
[3] FREERADIUS, Disponible en: <http://freeradius.org>