

Configuring EAP/TLS

Introduction

If there's one thing about wireless networking, previously, it's always been that there's no decent security. As it turned out WEP was a flop, there were known weaknesses from the start and was later rendered useless. Fortunately the 802.11i working group developed some more robust solutions. One of them is the EAP/TLS method of authentication which is based on existing standards.

EAP (Extensible Authentication Protocol) was originally developed with mainly PPP in mind, while TLS (the standard based on Netscape's SSL) was developed for Transport Layer Security, the handshake mechanism is a good means of authenticating. So TLS was adapted as one of the authentication methods in EAP, while EAP was adapted as EAPOL (EAP over LAN) for use with 802.11X to provide authentication on 802.3 LANs. Though enough of these tidbits about the various standards that form the base of EAP/TLS authentication on 802.11 and lets look at what it provides. First of all it provides reciprocal authentication, so not only is the client authenticated by the server, the client is also given the chance to authenticate the server. This is important because it prevents someone from setting up a rogue access point to attack the client. Furthermore we authenticate the client with an SSL client certificate. This is one sure way of preventing dictionary based attacks on weak passwords.

So providing your access point (AP) supports RADIUS authentication, as well as ensuring that your wireless device/computer supports certificate authentication, you should be able to use EAP/TLS authentication. Part of the authentication process is to agree on a key for the session to be used by both the authentication server (AS) and client. Fortunately this is handled by the TLS handshake and the AS sends the derived key on to the AP. From here the client and the AP authenticate each other before opening up an encrypted session. Here the options are TKIP or CCMP. TKIP (Temporal Key Integrity Protocol; rebranded as WPA - Wi-Fi Protected Access by the Wi-Fi Alliance), which is based on RC4 as used by WEP, was designed to be a secure intermediate solution to be available as an upgrade when WEP broke. However, as WEP hadn't broken when the 802.11i working group commenced, work was already underway on even more robust solution, CCMP (Counter-Mode/CBC-Mac Protocol or otherwise known as WPA2) which is based on AES. Although TKIP is considered secure, some inherent weaknesses meant that network disruptive countermeasures had to be used in case of attack. Therefore CCMP/AES is more desirable if available.

Now that you're through the primer, you'll probably want to get into it. At this stage this is not meant to be an exhaustive howto, though it's merely meant to build on what's available. So for a start you may like to take a look at the following documents:

- [EAPTLS.pdf](#) - covers configuration of FreeRADIUS and Windows XP supplicants (clients)
- [eap-tls HOWTO](#) - FreeRADIUS and XSupplicant

FreeRADIUS

If first thing's first it's probably configuring RADIUS. Just a note, RADIUS (Remote Authentication Dialin User Service) is another thing left over from the dialup days that has made its way into wireless networking. The reason being that it was quite an extensible protocol and there was room for EAP. The most common implementation under Linux is [FreeRADIUS](#). Hence, configuration for FreeRADIUS is covered.

Installation

I don't intend to cover installation in any great detail, however, if you're using [Debian](#) it's probably worth pointing out that the standard debian binary package doesn't include TLS support due to a licensing conflict. Therefore you'll need to download the [source](#) for FreeRADIUS. Once you have this, untar it and everything is there for you to build it into a debian package. For example, you'd run:

```
tar zxvf freeradius-1.0.1.tar.gz
cd freeradius-1.0.1/
dpkg-buildpackage -rfakeroot
```

If you don't have all the necessary developer (header file) packages, dpkg-buildpackage should inform you, however, you can also find what packages are required (build-dependencies) by looking in the *freeradius-1.0.1/debian/control* file. This should give you an installable debian package with EAP/TLS support.

Generating the certificates

Generating certificates with OpenSSL isn't always the most intuitive thing, yet it's one of the most fundamental part of any security system that uses SSL/TLS. The aforementioned howtos suggest you run an included script for this (though it was originally intended as a reference only). Although this might make it easier the first time, the script defaults to a weak passphrase. The least you can do for someone who steals your private key certificate is to make it easy for them to crack it by using a weak key. Also it's better to have some idea about what's going on so you are probably best to run the individual commands. If not, I suggest you change the included passphrase for each key. There are various programs in Linux that will help you generate a secure passphrase, so you might like to chose something like this 'YN4}6h<gdcv~s5[zr1"><'. Additionally, you'll want a different passphrase for each key. Furthermore by breaking this up into the various components, it should make it easier for you in understanding what you need to do for generating lots of client certificates, etc. For more details you might also like to read the [SSL Certificates HOWTO](#)</gdcv~s5[zr1">

No doubt, in most cases you'll want to be your own certificate authority. This basically involves self-signing your own root key that can then be used for signing client and server keys. Your organisation may already have root key for the purpose though. Other than that, the only reason I can think of where you may require the use of a recognised certification authority (CA) is if you're running a public hotspot. For internal use, it's just a matter of installing the root key on all your clients. To start out with, you'll probably want to choose a directory with only user permissions set (in debian you can probably use */etc/freeradius/certs* and it should not already contain a demoCA directory), then run the following commands to generate your self-signed root certificate (when prompted for a common name (CN), you can use anything, e.g. 'XYZ root certificate'):

CA root certificate

```
openssl req -new -x509 -keyout newreq.pem -out newreq.pem -days 730
echo "newreq.pem" | /usr/lib/ssl/misc/CA.pl -newca >/dev/null (location of CA.pl may vary)
openssl pkcs12 -export -in demoCA/cacert.pem -inkey newreq.pem -out root.p12 -cacerts
openssl pkcs12 -in root.p12 -out root.pem
openssl x509 -inform PEM -outform DER -in root.pem -out root.der
```

So now you have your root key, demoCA/cacert.pem will be used by FreeRADIUS to verify client keys, root.der is installed the supplicants to verify authentication servers, while root.p12, root.pem and demoCA/private/cakey.pem contain private keys used for signing other certificates and should be kept safe (possibly on a removable medium in case your server is ever compromised).

Next, we'll generate and sign the server certificate. Generally you only need to do this once, unless you have multiple radius servers on a large network. Note the XP client extensions, this is used so that XP clients can identify the certificate as an authentication certificate, so you'll need this [file](#). You'll also be prompted for your CA key passphrase when signing the newly certificate. When prompted for the common name (CN) this time, you can use the name of your authentication server or you could put 'XYD authentication server', it doesn't really matter except that the name should be unique for your network. One thing, you'll be prompted to enter 'A challenge password []:' and I'm not yet sure exactly what this is, though I suggest you leave it empty.

Server certificate

```
openssl req -new -keyout newreq.pem -out newreq.pem -days 730
openssl ca -policy policy_anything -out newcert.pem -extensions xpserver_ext -extfile
xpextensions -infile newreq.pem
openssl pkcs12 -export -in newcert.pem -inkey newreq.pem -out cert-srv.p12 -clcerts
openssl pkcs12 -in cert-srv.p12 -out cert-srv.pem
openssl x509 -inform PEM -outform DER -in cert-srv.pem -out cert-srv.der
```

There you have it, you now have the certificates required for FreeRADIUS. In particular it's cert-srv.pem that is used as the certificate for your server (note that it contains the server's private key). Last of all though, there is one other file that you'll need, that is the file for the Diffie-Hellman key agreement. To generate this key, go to /etc/freeradius/certs and run:

```
openssl dhparam -check -text -5 512 -out dh
```

Configuring FreeRADIUS

The following is an example of the configuration that's needed. In debian, most of it is located in /etc/freeradius/eap.conf:

```
eap {
    default_eap_type = tls
    ....
    tls {
        private_key_password = <passphrase that you entered for
server key>
```

```
private_key_file = ${raddbdir}/certs/cert-srv.pem
certificate_file = ${raddbdir}/certs/cert-srv.pem
CA_file = ${raddbdir}/certs/demoCA/cacert.pem
dh_file = ${raddbdir}/certs/dh
random_file = /dev/urandom
....
}
}
```

In addition, you'll need to check that `epap` is listed under `authorize { }` in the main `radius.conf` file. From here, you'll want to add your AP to the `clients.conf` file. Finally, your network is now ready to accept clients. You shouldn't need to add anything to the users database.

Configuring Clients

First things, first. You'll need to create a certificate for the client and sign it. The easiest way is to create it on the server (or the system where you copied the private CA key too). So to create the key with OpenSSL, run the following commands (we assume that you're in the same directory as where you created your CA/server keys). Once again you'll be prompted for the passphrase for the CA key. This time, when prompted for the common name (CN) this will be treated as the RADIUS User-Name attribute. Once again we also use the [xpextensions](#). For an XP supplicant, all you need is `cert-ct.p12` so you can probably stop after the third line.

Client certificate

```
openssl req -new -keyout newreq.pem -out newreq.pem -days 730
openssl ca -policy policy_anything -out newcert.pem -extensions xpclient_ext -extfile
xpextensions -infile newreq.pem
openssl pkcs12 -export -in newcert.pem -inkey newreq.pem -out cert-ct.p12 -clcerts
openssl pkcs12 -in cert-ct.p12 -out cert-ct.pem
openssl x509 -inform PEM -outform DER -in cert-ct.pem -out cert-ct.der
```

Now that you have your client certificate, you'll need to copy it to the client system. Remember that the `cert-ct.p12` file contains your private key so you'll want to do this over a secure connection (such as [pscp](#)). You'll also need to copy the CA public key, once again you'll want to ensure that this key hasn't been tampered with (as it's trusted by the client) so transfer it by a secure means.

Windows XP

To install the keys in Windows XP, follow the instructions as per [EAPTLS.pdf](#), however, I make the following difference. Instead of double clicking the root/CA key (`root.der`), I suggest you install it in Microsoft Management Console as Administrator by doing the following:

1. Add the snap-in for **Certificates (Local computer)**.
2. Expand **Trusted Root Certification Authorities**, right click **Certificates** and go to **All Tasks -> Import...**
3. Select your `root.der` file and follow the wizard.

Once done, the certificate is available on the whole system instead of just one account. Likewise, if you want network access before any users login, you can do the same by installing your client certificate (cert-clt.p12) into this location, but right clicking **Personal** and going to **All Tasks -> Import...** This time, however, you'll have to enter the passphrase. For some reason though, Windows insists that you install the key for every user (including your admin account). If you know a way around this, please let me know. It still works, but when you switch users it tries to re-authenticate and complains when it can't find a key for the user.

Now all you need to do is to configure your network device to use the keys. Go to **Network Connections**, right click your network device and go to properties and then the **Wireless Networks** tab. Select your network and go to properties then select the **Authentication** tab. Select EAP type: Smart Card or other Certificate and click properties. Make sure **Validate server certificate** is selected then scroll to your CA/root certificate and click it. Finally close all the dialogues and your computer should automatically connect to the network. If all goes well you'll be successfully authenticated and connected to your wireless network with EAP/TLS using the encryption protocol that you specified (TKIP or AES/CCMP).