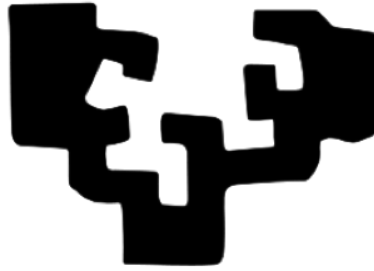


eman ta zabal zazu



The University of the Basque Country

---

Implementing IEEE 802.1X security  
for Ethernet networks on Linux

---

Author: Jorge Martínez de Salinas

Bilbao

July 1, 2009

# Contents

<b>1</b>	<b>IEEE 802.1X security</b>	<b>2</b>
1.1	Authentication server . . . . .	3
1.2	Authenticator . . . . .	4
1.3	Supplicant . . . . .	5

# List of Figures

1	IEEE 802.1X general architecture. . . . .	2
---	---	---

## 1 IEEE 802.1X security

The primary components of IEEE 802.1X include supplicants, authenticators, and authentication servers.

- **Supplicant.** A supplicant is a client device that needs to be authenticated before being allowed access to the network. Think of the supplicants as unknown users. Their identity is in question until they can produce valid credentials to the authentication server.
- **Authenticator.** An authenticator is a Layer 2 network device, such as an Ethernet switch or a wireless LAN access point. The authenticator acts as a security gate between the supplicants and the protected network. The gate (actually, port) stays closed until the authentication system verifies the credentials of the supplicant and deems that the supplicant is authorized to access the protected network. Once the system authenticates the supplicant, the authenticator will open a port so that the supplicant can access the protected network.
- **Authentication server.** The authentication server, for instance, will at some point request the credentials from the supplicant. The supplicant will then offer the credentials to the authentication server. The port-based authentication standards and specifications don't make any particular type of authentication server mandatory, but nearly all implementations utilize RADIUS.

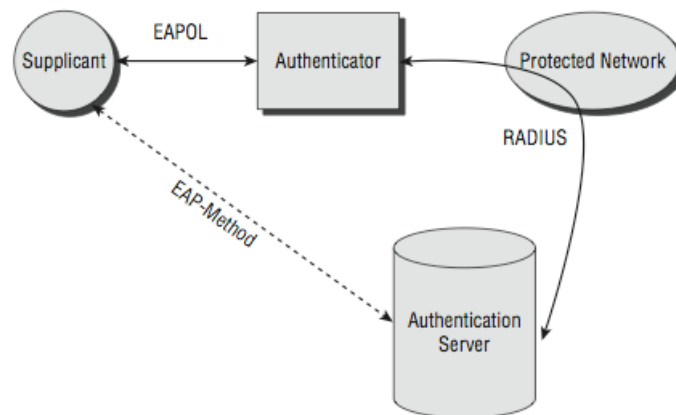


Figure 1: IEEE 802.1X general architecture.

## 1.1 Authentication server

On Linux systems the Authentication Server's function is performed by Freeradius [?], which can be easily installed via Yum.

```
# yum install freeradius
```

The Freeradius package included in Centos doesn't work with IPv6. If you need this functionality you need to compile Freeradius from source.

```
# ./configure && make && make install
```

At this point we have to decide which type of authentication will be using. In this case we've chosen EAP-TTLS/MD5. This method establishes a TLS tunnel based on digital certificates, and transports the MD5 authentication messages through this tunnel in a secure way.

First we need to modify `/etc/raddb/eap.conf`. Basically we need to uncomment the `tls` and `ttls` sections and specify the TLS configuration parameters such as CA's certificate, server's certificate and so on.

```
tls {
    private_key_password = secret
    private_key_file = ${raddbdir}/certs/spserver_key.pem
    certificate_file = ${raddbdir}/certs/spserver_cert.pem

    CA_file = ${raddbdir}/certs/cacert.pem

    dh_file = ${raddbdir}/certs/dh
    random_file = ${raddbdir}/certs/random
}

ttls {
    default_eap_type = md5
}
```

Next modify the `/etc/raddb/users` and add a new user:

```
jorge    User-Password := "jorge"
```

Freeradius is able to proxy EAP request to another Authentication Server. For instance, we can configure Freeradius to proxy the request from users belonging to a certain domain. To achieve it add the following to `proxy.conf`:

```
realm jorgemarsal.com {
    type                = radius
    authhost             = 10.0.1.100:1812
    accthost            = 10.0.1.100:1813
    secret              = testing123
    nostrip
}
```

## 1.2 Authenticator

The Authenticator function is performed by Hostadp. Depending on your Linux distribution, Hosapd packages may be available. Otherwise you can download the sources and compile them.

```
# wget http://hostap.epitest.fi/releases/hostapd-0.6.9.tar.gz
# tar xfvz hosapd-0.6.9.tag.gz
# cd hostapd-0.6.9/hostapd/
```

You can choose which modules are going to be compiled tweaking .config file. Just be sure to uncomment CONFIG\_DRIVER\_WIRED=y. defconfig file includes all the configuration options.

```
# cp defconfig .config
# make
# make install
```

Next edit the hosapd.conf:

```
interface=eth0
driver=wired

own_ip_addr=10.0.0.2

# RADIUS authentication server
auth_server_addr=10.0.0.100
auth_server_port=1812
auth_server_shared_secret=secret

# RADIUS accounting server
acct_server_addr=10.0.0.100
acct_server_port=1813
acct_server_shared_secret=secret
```

To start hostapd type:

```
# hostapd -dd hostapd.conf
```

## 1.3 Supplicant

The supplicant SW is made by the same author of the Authenticator so the compilation steps are very similar. The Supplicant function is performed by wpa\_supplicant. Depending on your Linux distribution, wpa\_supplicant packages may be available. Otherwise you can download the sources and compile them.

```
# wget http://hostap.epitest.fi/releases/wpa_supplicant-0.6.9.tar.gz
# tar xfvz wpa_supplicant-0.6.9.tar.gz
# cd wpa_supplicant-0.6.9/wpa_supplicant/
```

You can choose which modules are going to be compiled tweaking .config file. Just be sure that CONFIG\_EAP\_TLS and CONFIG\_EAP\_TTLS are both set to yes.

```
# cp defconfig .config
# make
# make install
```

Next edit wpa\_supplicant.conf.

```
ctrl_interface=/var/run/wpa_supplicant
ctrl_interface_group=0
ap_scan=0
network={
    key_mgmt=IEEE8021X
    eap=TTLS
    identity="jorge@jorgemarsal.com"
    anonymous_identity="anonymous@example.com"
    password="jorge"
    ca_cert="/etc/cert/ca.pem"
    priority=2
    eapol_flags=0
}
```

To start wpa\_supplicant type:

```
# wpa_supplicant -dd -Dwired -ieth0 -c ./wpa_supplicant.conf
```

```
[root@AN hostapd]# ./hostapd hostapd.conf
Configuration file: hostapd.conf
Using interface eth0 with hwaddr 00:0c:29:b4:be:e2 and ssid ''
eth0: RADIUS Authentication server 10.0.0.100:1812
eth0: RADIUS Accounting server 10.0.0.100:1813
eth0: STA 00:0c:29:ff:4b:37 RADIUS: starting accounting session \
4A3542A4-00000000
eth0: STA 00:0c:29:ff:4b:37 IEEE 802.1X: authenticated \
- EAP type: 21 (TTLS)
```

```
[root@Customer wpa_supplicant]# wpa_supplicant -i eth0 \
-Dwired -c wpa_supplicantttls.conf
Device eth0 kernel driver name: pcnet32.
Associated with 01:80:c2:00:00:03
CTRL-EVENT-EAP-STARTED EAP authentication started
CTRL-EVENT-EAP-METHOD EAP vendor 0 method 21 (TTLS) selected
CTRL-EVENT-EAP-SUCCESS EAP authentication completed successfully
CTRL-EVENT-CONNECTED - Connection to 01:80:c2:00:00:03 \
completed (auth) [id=0 id_str=]
```