TRIP - Implementazione Tecnica

Riccardo.Veraldi@cnaf.infn.it
8/5/2006

Lo scopo del progetto è di implementare un'architettura software e hardware per consentire ad un utente che deve accedere alla rete INFN WiFi, un'autenticazione facile svincolata dalla struttura ospitante. L'autenticazione è riguarda due differenti tipologie di utenti: utenti INFN (dipendenti e associati) e utenti non INFN, cioè ospiti appartenenti ad alter strutture.
Per quanto riguarda gli utenti INFN l'autenticazione si basa su EAP-TTLS, utilizzando un'architettura di server radius distribuiti.
Per gli utenti ospiti l'autenticazione è basata sull'utilizzo di un portale web.
Diseguito sarà analizzato in dettaglio l'implementazione della struttura di TRIP.

Requisiti Hardware

WiFi access point
E necessario l'utilizzo di accesss point wireless che riescano a gestire SSID multipli e le VLAN. Ogni SSID sarà associato a una diversa VLAN. Inoltre l'access point dovrà essere in grado di supportare WPA e WPA2 e quindi i metodi di encryption AES e TKIP per avere pieno supporto con lo standard 802.11i
Abbiamo utilizzato gli access point Cisco 1200 (AIR-AP1231G-E-K9), che possono essere gestiti tramite interfaccia Cisco IOS oppure tramite web.

2 Nodi rack 1U
È necessario l'utilizzo di una macchina adibita a radius server, e di una macchina adibita al servizio di portale web. Per entrambe si sono utilizzate dei nodi 1U con biprocessori Xeon a 3GHz.
Il sistema operativo utilizzato è Scientific Linux 4.x

Switch di rete
È necessario l'utilizzo di switch di rete di layer2/3 che sappiano gestire le VLAN e il VLAN tagging. In particolare abbiamo utilizzato switch Cisco 3750 ed Extreeme Networks Summit 400i. L'utilizzo di questi con l'acccess point Cisco 1200 richiede che lo switch di rete sia in grado di gestire la VLAN 1 in modalità untagged.

Architettura TRIP

L'architettura di TRIP si basa su un modello distribuito, utilizzando SSID multipli.
In particolare si utilizza l'SSID **INFN-dot1x** per gli Utenti INFN, e l'SSID **INFN-Web**, per gli utenti non INFN.
L'utente INFN che si trova in una qualsiasi sede INFN si associa all'access point utilizzando **INFN-dot1x** come SSID e EAP-TTLS come protocollo di autenticazione. Questo consente il tunneling delle credenziali all'interno di un canale cifrato TLS.
Se l'utente è un utente locale, l'autenticazione viene effettuata dal server radius della sezione, alternativamente la sua autenticazione viene demandata al server radius della

propria sede tramite il meccanismo proxy radius. Le credenziali verranno controllate dal radius server, la risposta arriverà all'access point il quale consentirà o negherà l'accesso alla rete (fig.1).

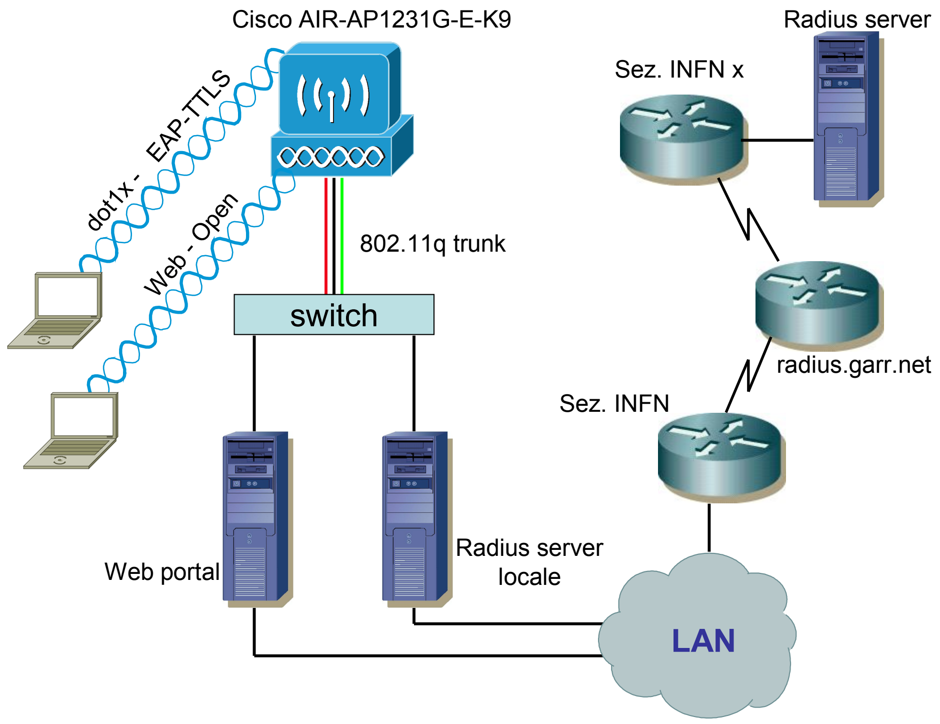


fig.1

L'architettura quindi si basa sul modello di autenticazione radius proxy distribuita nelle diverse sezioni. Ogni sezione potrà decidere di utilizzare il database più opportuno per la gestione delle credenziali per i propri utenti. Sono stati testati due scenari che possono essere i due più comunemente usati. L'utilizzo di un database radium gestito localmente con file di password unix o file di testo *users*, e l'integrazione fra radius server e autenticazione Kerberos con autorizzazione LDAP.

Configurazione Cisco AP 1200

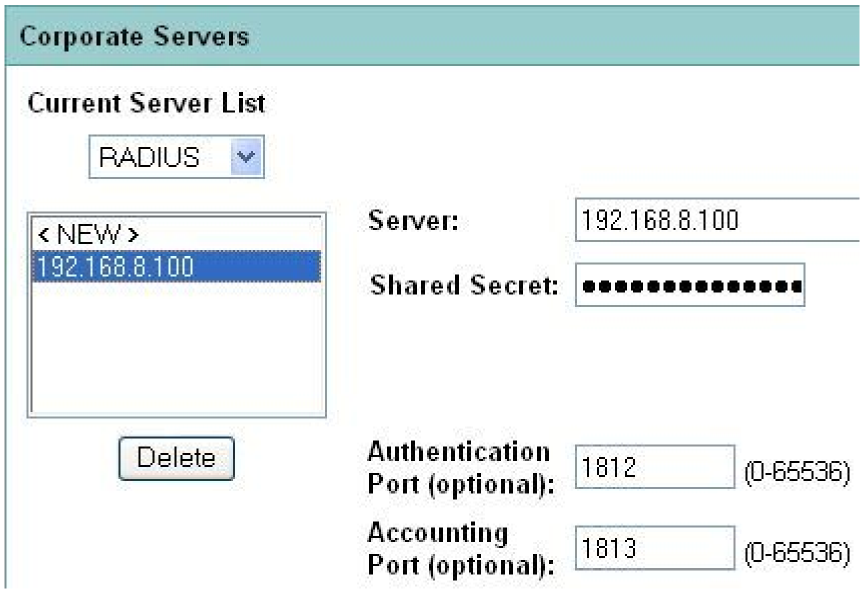
È consigliabile installare l'ultima versione di IOS disponibile.
L'installazione può essere effettuata tramite interfaccia WEB selezionando *System Software* e successivamente *Software Upgrade*. Alternativamente si può installare tramite interfaccia IOS con il comando:

```
archive download-sw /overwrite tftp://server-tftp
```

La configurazione dell'access point viene effettuata via WEB.
La parte più rilevante riguarda la configurazione della sezione *Security*.

Inizialmente dobbiamo definire un radius server alla voce *Server Management*. Lo shared Secret richiesto dovrà successivamente essere configurato all'interno del radium server

stesso. Le porte (UDP) da utilizzare sono 1812 e 1813 come mostrato in figura 2.
L'indirizzo IP del radius server va inserito anche nella voce relativa a *EAP Authentication*.


Corporate Servers

Current Server List

RADIUS

< NEW >
192.168.8.100

Server: 192.168.8.100

Shared Secret: ●●●●●●●●●●●●●●

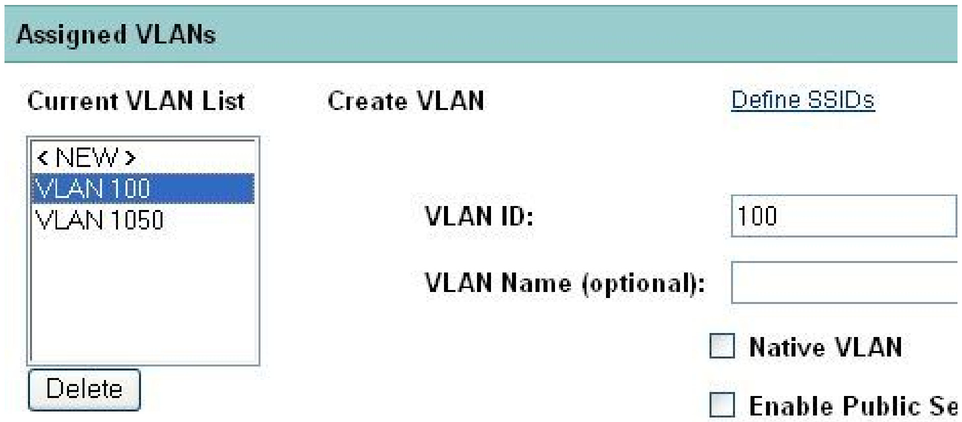
Delete

Authentication Port (optional): 1812 (0-65536)

Accounting Port (optional): 1813 (0-65536)

fig2

Successivamente occorre selezionare *SSID Manager* e poi *Define VLANS*. Bisogna definire 2 diverse VLAN, ad esempio VLAN 100 per gli utenti 802.1x e VLAN 1050 per gli utenti che afferiscono al portale WEB.


Assigned VLANs

Current VLAN List

Create VLAN

Define SSIDs

< NEW >
VLAN 100
VLAN 1050

VLAN ID: 100

VLAN Name (optional):

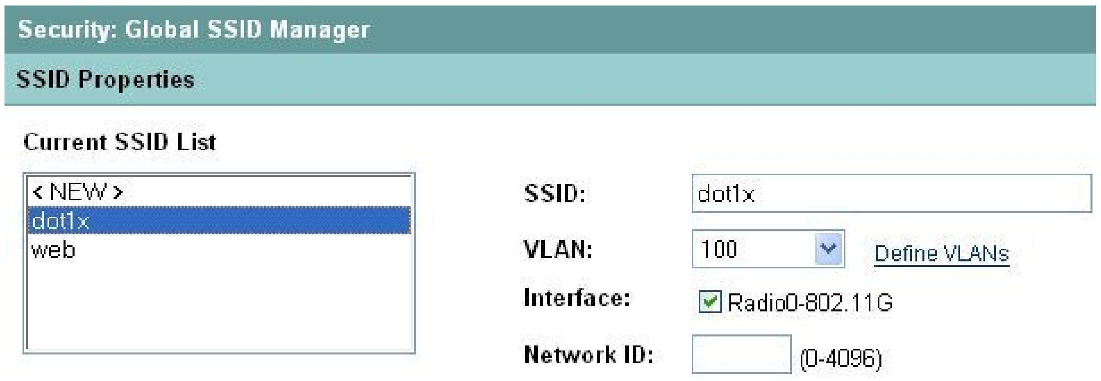
☐ Native VLAN

☐ Enable Public Se

Delete

fig.3

A questo punto si definiscono due SSID che chiameremo **INFN-dot1x** e **INFN-Web**. Assegnamo il primo alla VLAN 100 e il secondo alla VLAN 1050, ed associamo entrambi all'interfaccia Radio0-802.11G come mostrato in figura 4.


Security: Global SSID Manager

SSID Properties

Current SSID List

< NEW >
dot1x
web

SSID: dot1x

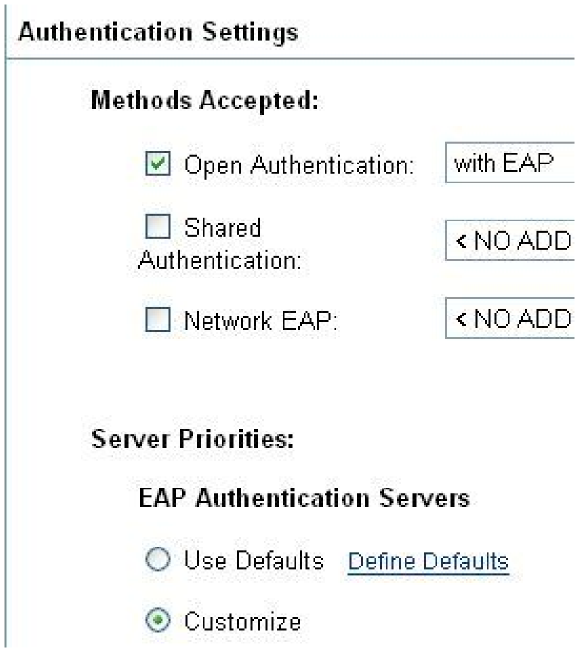
VLAN: 100 Define VLANs

Interface: ✓ Radio0-802.11G

Network ID: (0-4096)

fig.4

Per l’SSID **INFN-Web** Si seleziona *Open Authentication* senza nessuna ulteriore specifica, per **INFN-dot1x** selezioniamo *Open Authentication with EAP*.


Authentication Settings

Methods Accepted:

✓ Open Authentication: with EAP

☐ Shared Authentication: < NO ADD

☐ Network EAP: < NO ADD

Server Priorities:

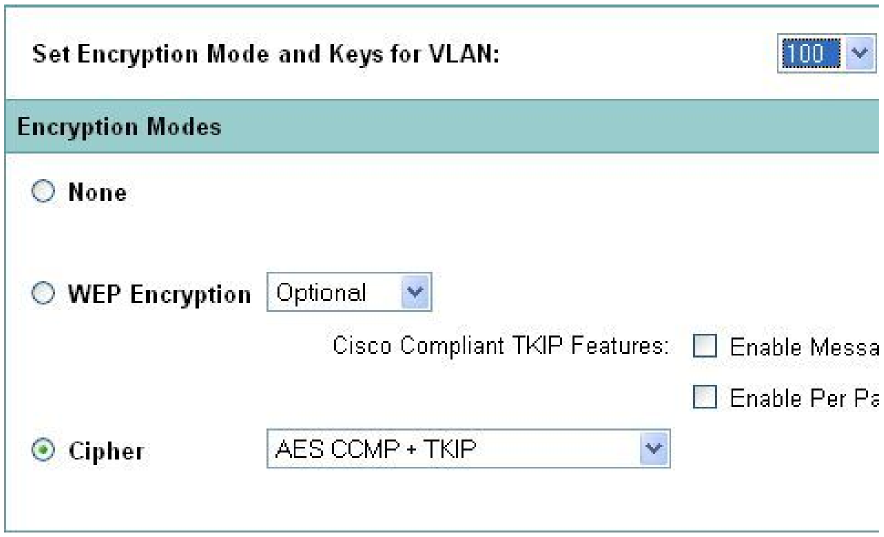
EAP Authentication Servers

○ Use Defaults Define Defaults

● Customize

fig.5

Per quanto riguarda l’SSID **INFN-Web** non selezioniamo nessun tipo di key management.
A questo punto nella parte *Encryptin Manager* selezioniamo inizialmente la VLAN 1050 relativa al SSID **INFN-Web** e come *Encryption mode* si seleziona *none*. Successivamente selezioniamo la VLAN 100 e come *Encryption mode* si setta il *chiper AES-CCMP+TKIP* (fig.6).



Set Encryption Mode and Keys for VLAN: 100

Encryption Modes

○ None

○ WEP Encryption Optional

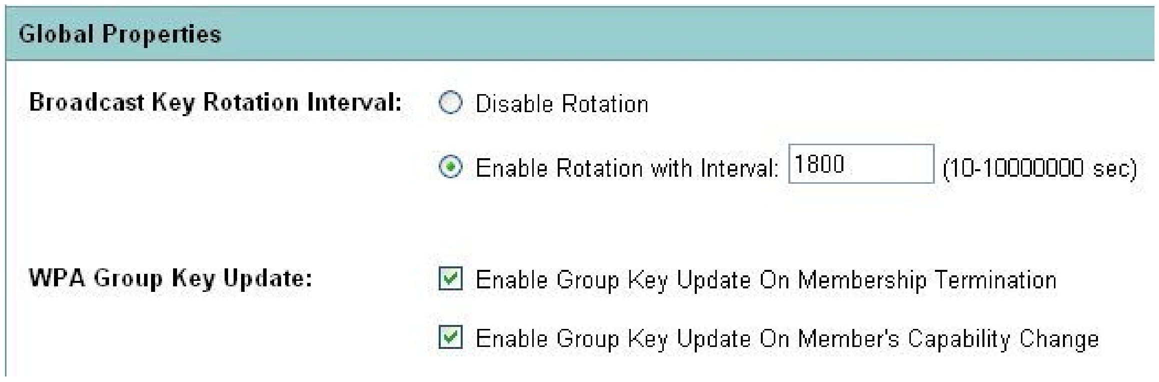
Cisco Compliant TKIP Features: ☐ Enable Messa

☐ Enable Per Pa

◉ Cipher AES CCMP + TKIP

fig.6

In questo modo supporteremo la modalità mista per entrambi i supplicant compliant con WPA e WPA2. È poi necessario abilitare la rotazione delle chiavi WPA con l'opzione *Broadcast Key Rotation Interval* definendo un intervallo di tempo, come mostrato in figura 7.



Global Properties

Broadcast Key Rotation Interval: ○ Disable Rotation

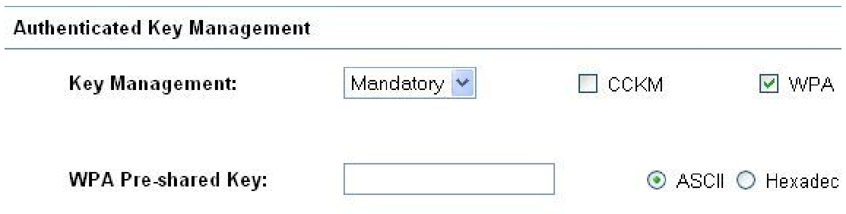
◉ Enable Rotation with Interval: 1800 (10-10000000 sec)

WPA Group Key Update: ☑ Enable Group Key Update On Membership Termination

☑ Enable Group Key Update On Member's Capability Change

fig.7

Tornando ora in *SSID Manager* si seleziona il SSID **INFN-dot1x** e nel campo *Key management* selezioniamo *Mandatory* con *WPA* abilitato come mostrato in figura 8. Si può poi decidere di propagare uno dei due o entrambi gli SSID in modo che siano visibili ai client. La soluzione migliore rimane comunque quella di annunciare i SSID in broadcast per ragioni di sicurezza.



Authenticated Key Management

Key Management: Mandatory ☐ CCKM ☑ WPA

WPA Pre-shared Key: ◉ ASCII ○ Hexadec

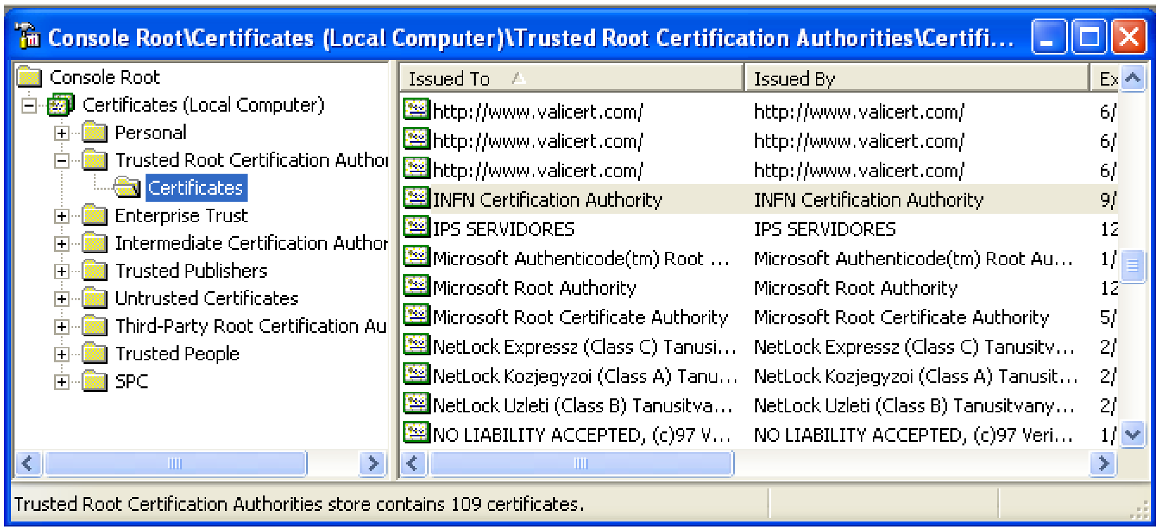
fig.8

Configurazione Client

La configurazione dei client è stata testata su sistemi Windows XP, Linux Scientific Linux 4.x.

Certificato INFN-CA

Per le operazioni successive è necessario importare il certificato della INFN-CA all’interno del Microsoft store relativo al computer. Bisogna lanciare il comando **mmc**.


Console Root\Certificates (Local Computer)\Trusted Root Certification Authorities\Certifi...

Console Root
- Certificates (Local Computer)
  - Personal
  - Trusted Root Certification Author
    - Certificates
  - Enterprise Trust
  - Intermediate Certification Author
  - Trusted Publishers
  - Untrusted Certificates
  - Third-Party Root Certification Au
  - Trusted People
  - SPC

Issued To	Issued By	Ex
http://www.valicert.com/	http://www.valicert.com/	6/
http://www.valicert.com/	http://www.valicert.com/	6/
http://www.valicert.com/	http://www.valicert.com/	6/
INFN Certification Authority	INFN Certification Authority	9/
IPS SERVIDORES	IPS SERVIDORES	12
Microsoft Authenticode(tm) Root ...	Microsoft Authenticode(tm) Root Au...	1/
Microsoft Root Authority	Microsoft Root Authority	12
Microsoft Root Certificate Authority	Microsoft Root Certificate Authority	5/
NetLock Expressz (Class C) Tanusi...	NetLock Expressz (Class C) Tanusitv...	2/
NetLock Kozjegyzoi (Class A) Tanu...	NetLock Kozjegyzoi (Class A) Tanusit...	2/
NetLock Uzleti (Class B) Tanusitva...	NetLock Uzleti (Class B) Tanusitvany...	2/
NO LIABILITY ACCEPTED, (c)97 V...	NO LIABILITY ACCEPTED, (c)97 Veri...	1/

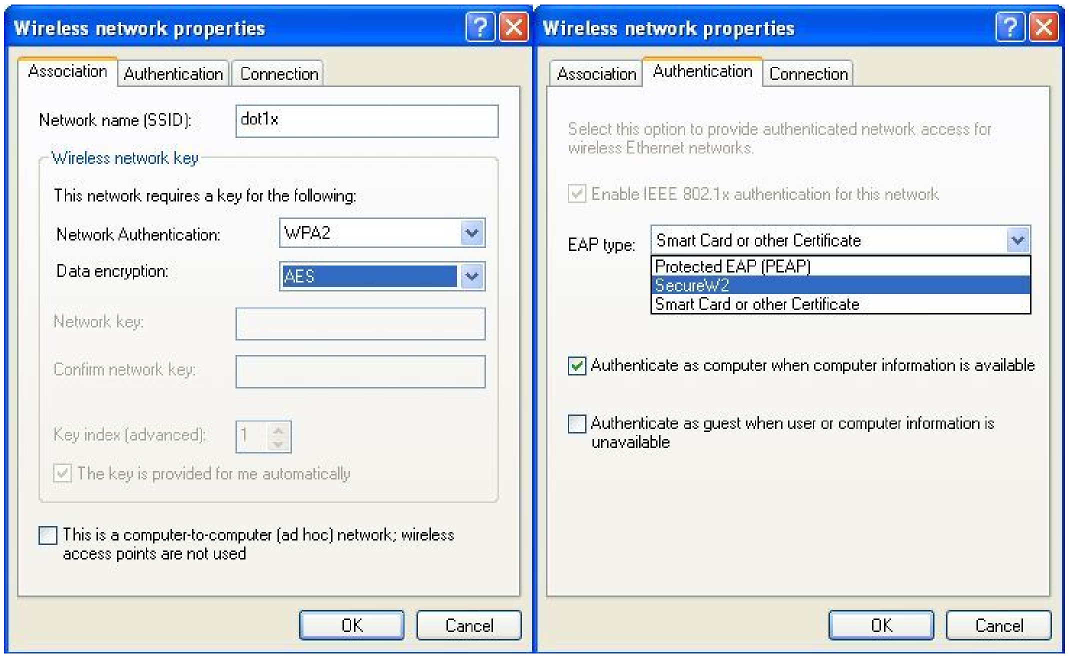
Trusted Root Certification Authorities store contains 109 certificates.

fig.9

Da qui si seleziona **Add/Remove Snap-in** e si aggiunge la parte relativa ai certificati installati nel computer locale, e si aggiunge INFN-CA all’interno delle trusted Root Certification Authorities, importando il certificato.
Alla fine si ottiene la schermata come in figura
Il certificato dell’INFN-CA si scarica da https://security.fi.infn.it/CA/mgt/getCA.php

Configurazione Windows XP SP2 per gli utenti strutturati INFN

È opportuno aggiornare la versione del driver della propria scheda WiFi in modo da supportare WPA. Le schede più datate non supportano questa opzione cherichiede un hardware specifico. Sarà quindi necessario dotarsi di hardware più recente.
Inoltre bisogna installare l’update KB893357 di microsoft relativo al supporto dello standard 802.11i per avere supporto per WPA2.
Per la tipologia utente INFN si utilizza EAP-TTLS. È necessario installare il driver che non è presente nella distribuzione di Windows XP. Si può scaricare da:
http://security.fi.infn.it/TRIP/sw/win/SecureW2_312.exe
Dopo avere installato questi componenti si può procedere alla configurazione del client come in figura 10.



Wireless network properties

Association | Authentication | Connection

Network name (SSID): dot1x

Wireless network key

This network requires a key for the following:

Network Authentication: WPA2

Data encryption: AES

Network key:

Confirm network key:

Key index (advanced): 1

☑ The key is provided for me automatically

☐ This is a computer-to-computer (ad hoc) network; wireless access points are not used

Wireless network properties

Select this option to provide authenticated network access for wireless Ethernet networks.

☑ Enable IEEE 802.1x authentication for this network

EAP type: Smart Card or other Certificate

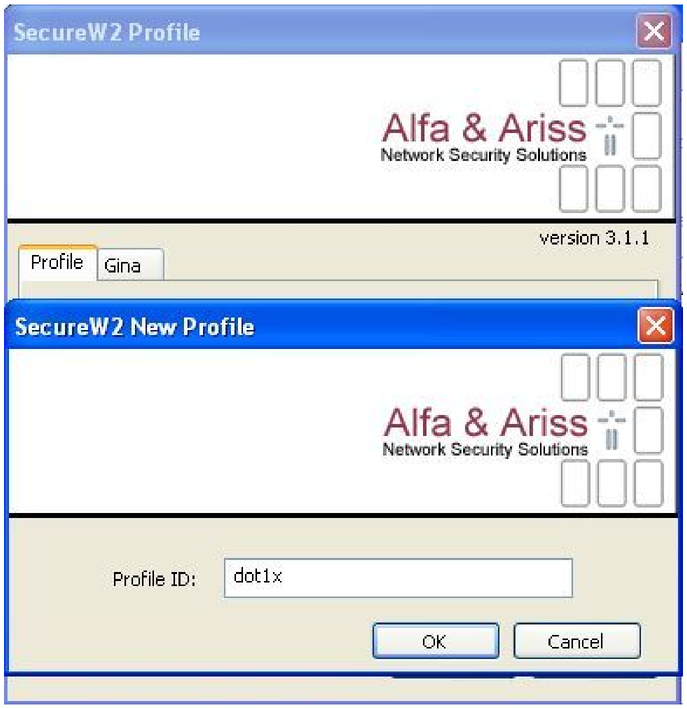
Protected EAP (PEAP)
SecureW2
Smart Card or other Certificate

☑ Authenticate as computer when computer information is available

☐ Authenticate as guest when user or computer information is unavailable

fig.10

.Si definisce il SSID **INFN-dot1x.** Si seleziona ove possibile *WPA2* come network authentication e *AES* come data encryption. Alternativamente, se WPA2 non è supportato dalla propria scheda, occorre specificare *WPA* con *TKIP* come encryption. A questo punto si seleziona *SecureW2* nella finestra *Authentication*. In questo modo verrà utilizzato il modulo EAP-TTLS di Alfa-Ariss che consente di inviare le informazioni relative alle credenziali (login e password) all'interno di un tunnel TLS.



SecureW2 Profile

version 3.1.1

Profile | Gina

SecureW2 New Profile

Profile ID: dot1x

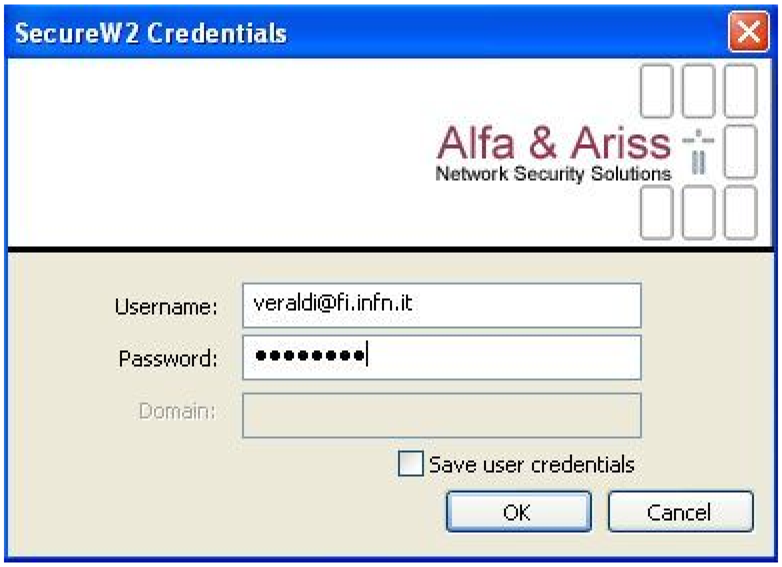
OK | Cancel

fig.11

Si crea un nuovo profilo **INFN-dot1x** come mostrato in figura 11 e si definiscono le proprietà con il tasto *configure*.

- Nella parte relativa a *Connection* bisogna disabilitare la voce *Use alternate outer Identity*.
- Alla voce *Certificates* occorre selezionare il certificato dell'INFN-CA in *Add-CA*.
- Alla voce Authentication è possibile specificare *EAP-MD5-Challenge* nel caso in cui l'autenticazione radius sia effettuata localmente in un database gestito dal radius server stesso (file di password, ad esempio **`/etc/raddb/users`**). Nel caso di qualsiasi altro genere di autenticazione si deve specificare PAP.
- Alla voce *User Account* si può specificare *Prompt user for credentials*, oppure le credenziali possono essere inserite direttamente nella finestra.

Si salva il profilo e ci si collega alla rete appena configurata come mostrato in figura 12.



Username: veraldi@fi.infn.it
Password: ●●●●●●●●●
Domain:
[ ] Save user credentials

fig.12

Si inseriscono le credenziali, e l'autenticazione verrà demandata al server radius che appartiene al dominio specificato. Nel caso il dominio sia locale allora l'autenticazione sarà effettuata dal server radius locale.

Configurazione Client Scientific Linux 4.1 per gli utenti strutturati INFN

La configurazione è stata testata su piattaforma Scientific Linux 4.1 su portatile IBM Thinkpad. La scheda WiFi utilizzata è stata la Cisco Aironet 802.11.a/b/g che ha Chipset Atheros.
È necessario innanzitutto installare le seguenti rpm presenti nella distribuzione di Scientific Linux:

- kernel-2.6.9-22.0.1.EL
- wireless-tools-27-0.pre25.4.EL4

Il secondo pacchetto comprende I tool necessary per il management della scheda WiFi come *iwconfig, iwpriv, iwlist*.
Occorre poi installare il driver opportuno che supporti il Chipset Atheros ed 802.1x. Questo si scarica da http://atrpms.net/dist/el4/madwifi/. La rpm è quella relativa alla propria versione di kernel quindi in questo caso *madwifi-kmdl-2.6.9-22.0.1.EL-0.9.6.0-18.el4.at.*
Bisogna poi installare rispettivamente da http://atrpms.net/dist/el4/pcsclite/ e da http://atrpms.net/dist/el4/wpa_supplicant/ i pacchetti wpa*_supplicant-0.3.9-8.el4.at* e *libpcsclite0-1.2.0-5.el4.at*.
Wpa_supplicant è il software che serve per accedere alla rete WiFi utilizzando WPA o WPA2 ed EAP-TTLS.
Il file di configurazione è *wpa_supplicant.conf*. Di seguito è riportata la configurazione EAP-TTLS:

wpa_supplicant.conf:

```
network={
     ssid="INFN-dot1x"
     proto=RSN WPA
     scan_ssid=1
     key_mgmt=WPA-EAP
     pairwise=CCMP TKIP
     group=CCMP TKIP
     eap=TTLS
     identity="username"
     ca_cert="/path/to/ca/certificate"
     password="********"
     phase2="auth=PAP"
}
```

In particolare si specifica il protocollo da utilizzare WPA2 (RSN) oppure WPA, il tipo di key management, gli algoritmi di encryption AES-CCMP oppure TKIP, il tipo EAP di default (TTLS), l'username, la password dell'utente e infine il tipo di autenticazione all'interno del tunnel TLS. Se non si specifica alcun tipo il default utilizzato e' EAP-MD5. Nel caso si utilizzi Kerberos per l'autenticazione a livello di radius server occorre utilizzare PAP nella phase2

Una volta configurato *wpa_supplicant* bisogna dare il comando:

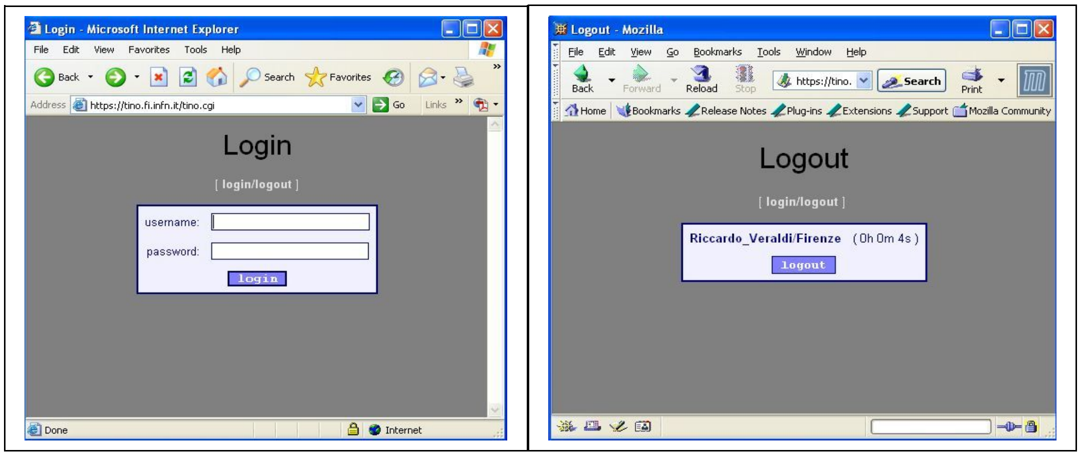
```
wpa_supplicant -d -w -iath0 -Dmadwifi -c /etc/wpa_supplicant.conf
```

Il client si associa all'access point previa autenticazione con il radius server.
Ad associazione avvenuta si può attivare al scheda di rete WiFi **ath0** per acquisire un indirizzo IP tramite dhcp:

```
/sbin/dhclient ath0
```

Configurazione per utenti non strutturati INFN (Ospiti)

Per utenti non INFN, e quindi ospiti è previsto l'accesso alla rete WiFi tramite portale Web. È sufficiente configurare il proprio client con SSID **INFN-Web**, network authentication *Open* con nessun tipo di encryption. A questo punto, una volta aperto il proprio browser Web si viene ridirezionati sulla pagine di login del portale WEB. Se nel proprio browser è installato un certificato digitale sarà tentata l'autenticazione con certificato, altrimenti verrà richiesto username e password come mostrato in figura.


Login - Microsoft Internet Explorer
https://tino.fi.infn.it/tino.cgi
Login
[login/logout]
username:
password:
login
Logout - Mozilla
Logout
[login/logout]
Riccardo_Veraldi/Firenze (0h 0m 4s)
logout
fig. 13

Configurazione 802.1x per server radius

L'architettura TRIP si basa su una rete di server radius distribuiti, con un server radius centrale (radius.garr.net) che ha il compito di smistare le richieste entranti e di indirizzarle al radius server corretto a seconda del dominio di appartenenza della richiesta.
Il radius server hub in questione è basato sul software *radiator*.
Nella fase di sperimentazione si è deciso di utilizzare Freeradius sui radius server delle diverse sezioni. È un software freeware, ma supporta diversi meccanismi di autenticazione, e il suo utilizzo negli ultimi due anni ha consentito di avere una conoscenza buona del prodotto. Di seguito saranno riportate le configurazioni di freeradius. Il sistema su cui si è installato freeradius è *Scientific Linux 4.1* con kernel *2.6.9-22.0.1.ELsmp*. La versione di freeradius utilizzata è quella resa disponibile nella distribuzione *Scientific Linux* nel pacchetto *rpm freeradius-1.0.1-3.RHEL4*.

La parte di configurazione rilevante riguarda i file *radiusd.conf*, *eap.conf*. *clients.conf*, *proxy.conf*, *users*.

clients.conf: in questo file di configurazione vanno riportati gli indirizzi IP dei NAS che accedono al radius server per autenticare i supplicant. Vanno pertanto inseriti gli indirizzi

IP degli access point e lo shared secret che condividono con il radius server. Va anche inserita la stanza che riguarda l'accesso al radius server hub. Di seguito viene riportata una configurazione di esempio:

```
# clients.conf
# access point 1
client 172.16.0.1 {
   secret  = ********
   shortname = cisco1200
}
#acces point 2
   client 172.16.0.2 {
   secret  = ********
   shortname = cisco1200
}
#
client 192.84.145.15 {
   secret = ********
   shortname = radiushub
}
```

proxy.conf: in questo file risiede la configurazione del proxying di radius. Se il realm del supplicant che si autentica corrisponde a quello locale verrà utilizzato il radius server locale altrimenti l'autenticazione sarà rimandata al radius server hub:

```
realm LOCAL {
   type = radius
   authhost = LOCAL
   accthost = LOCAL
}

realm dominio.infn.it {
   type = radius
   authhost = LOCAL
   accthost = LOCAL
}

realm DEFAULT {
   type = radius
   authhost = radius.garr.net:1812
   accthost = radius.garr.net:1813
   secret   = ********
   nostrip
}
```

eap.conf: in questo file risiede la configurazione EAP. È importante specificare

```
default_eap_type = ttls
```

come attributo EAP di default, in modo che sia utilizzato EAP-TTLS. Si può utilizzare in questo modo contemporaneamente EAP-TTLS e EAP-PEAP.

radiusd.conf: in questo file risiede la configurazione globale del server radius. La configurazione di questo file è molto critica. La parte più importante riguarda le sezioni *authorize* ed *auhtenticate*. Nel caso si voglia utilizzare Kerberos come meccanismo di autenticazione è necessario aggiungere nella sezione *authenticate* la seguente stanza:

```
Auth-Type Kerberos {
   krb5
}
```

Nella sezione *modules* bisogna aggiungere:

```
krb5 {
        keytab = /etc/krb5.keytab
}
```

In Appendice A sono riportati due file di configurazione *radiusd.conf* relativi ad una configurazione con autenticazione Kerberos - autorizzazione LDAP, e più semplicemente con autenticazione su database locale (unix password). Queste configurazioni rappresentano due possibili scenari diversi.

Users: in questo file si configurano i tipi di autenticazione permessi, associati eventualmente con uno username, e con proprietà specifiche. Più comunemente conviene configurare un tipo di autenticazione di DEFAULT per tutti gli utenti, ad esempio Kerberos oppure System (unix password), e disabilitare invece l'autenticazione tramite certificato digitale. In appendice sono riportati due possibili scenari di configurazione.

Captive Portal

Il captive portal consente di convogliare l'autenticazioen WiFi a livello utente su una pagina web che chiede le credenziali di login. Una volta che l'utente si sarà autenticato, verrà consentito l'accesso alla rete.
La soluzione studiata si basa sul portale web TINO sviluppato al politecnico di Vaasa (Finlandia). Il pacchetto è stato modificato all'INFN di Firenze per aggiungere il supporto con l'autenticazione tramite i certificati digitali.

Principio di funzionamento di TINO Captive Portal

Quando un utente apre il proprio browser web viene ridirezionato sulla pagina web di login di TINO. Vengono analizzati i mac address presenti nel dhcp lease file. All'utente viene richiesta l'autenticazione tramite certificato digitale o tramite login e password. Successivamente, se l'utente si è autenticato, il nome dell'utente, l'indirizzo IP e il mac address vengono salvati in un file nell'area di spooling. TINO chiama uno script di firewall per aprire la connessione all'utente in base all'indirizzo IP e/o mac address. Periodicamente tramite uno script girato da cron, TINO controlla gli utenti che sono autenticati con le informazioni presenti nell'area di spooling e nel dhcp lease time. Se il dhcp lease time è scaduto, TINO rimuove la firewall entry e cancella il file con le informazioni dall'area dispooling.

Installazione di TINO.

Il sistema è stato testato su Scientific Linux 4.2.

1. Scaricare i pacchetti necessari all'installazione di TINO da http://security.fi.infn.it/TRIP/sw/linux
   In particolare oltre che alla distribuzione di TINO sono necessarie le rpm relative a perl-Authen-Radius-0.12 , perl-Crypt-PasswdMD5-1.3 , perl-Data-HexDump-0.02.
2. copiare l'archivio tar di TINO in /usr/local e scompattarlo.
   Verrà creata una directory /usr/local/tino che contiene tutta la distribuzione.
3. Creare una directory /usr/local/tino-www e settare le ownership in modo che l'utente del server WEB sia il proprietario.
4. Copiare tutti i file html e css presenti in /usr/local/tino in /usr/local/tino-www e nel caso sia necessario personalizzarli
5. editare il file tino.pm in /usr/local/tino e personalizzarlo secondo le proprie esigenze

L'installazione di TINO richiede la configurazione del server apache. La versione utilizzata è la 2.0.52-22 del pacchetto rpm fornito con Scientific Linux 4.2

La parte più rilevante riguarda la configurazione in httpd.conf per ridirezionare le URL web sul portale alla porta SSL wrapped 443 su https:

```
RewriteEngine on
RewriteRule ^(.*) https://tino.sezione.infn.it [R=301,QSA,L]
```

Inoltre nella configurazoine in ssl.conf bisogna abilitare il server sulla porta 443 e specificare le opzioni relative alla gestione delle variabile di ambiente per i certificati digitali. Inoltre bisogna abilitare l'esecuzione della parte cgi di TINO.

```
 <VirtualHost _default_:443>

# General setup for the virtual host, inherited from global configuration
DocumentRoot "/usr/local/tino-www"

# tino config
```

```
ScriptAlias /tino.cgi "/usr/local/tino/tino.cgi"

<Location /tino.cgi>
 SSLVerifyClient optional
 SSLOptions +StdEnvVars
 SSLOptions +ExportCertData
 SSLVerifyDepth  2
</Location>
```

firewall.sh

è necessario scrivere uno script di firewall che di default chiuda le connessioni, ma che le apra per un determinato IP relativo all’utente che si autentica con successo. Lo script di seguito utilizza iptables

```
#!/bin/sh

if [ $1 =  open ]; then
        iptables -t nat -I PREROUTING -i eth1 -p tcp -s $2 -j ACCEPT
        iptables -I FORWARD -i eth1 -s $2 -j ACCEPT
fi

if [ $1 = close ]; then
        iptables -t nat -D PREROUTING -i eth1 -p tcp -s $2 -j ACCEPT
        iptables -D FORWARD -i eth1 -s $2 -j ACCEPT
fi

if [ $1 = reset ]; then
        /etc/init.d/iptables restart
fi
```

Questo script viene eseguito ad ogni login e ad ogni logout dell’utente, e quando scade il tempo di timeout per la connessione. I client sono su una sottorete nascosta che viene nattata sull’interfaccia esterna (eth0) del portale web. Le regole di filtro vengono inserite relativamente all’interfaccia interna (eth1) nella coda di FORWARD del traffico.
L'IP forwarding deve essere abilitato inserendo la riga

```
net.ipv4.ip_forward = 1
```

in **`/etc/sysctl.conf`** [rc – 5/4/06]. Come soluzione di emergenza senza far ripartire il sistema:

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

È possible fare il logging delle sessioni del portale web in un file ad esempio `/var/log/tino`

```
2006.01.26 19:05:41 Login: veraldi@cnaf.infn.it 172.16.30.253 00:40:96:a7:11:ba
2006.01.26 19:05:50 Logout: veraldi@cnaf.infn.it 172.16.30.253 00:40:96:a7:11:b
```

Viene registrato il MAC address l’indirizzo IP associati all’utente.

Appendice A

Di seguito vengono riportate le configurazioni *freeradius* per due scenari che sono stati testati, e che possono essere i più comunemente utilizzati nell'ambito INFN:

- Autenticazione kerberos e autorizzazione ldap
- Autenticazione locale con file di password unix

In entrambi i casi il file di configurazione *eap.conf* rimane inalterato.
La configurazione di seguito setta come default EAP type TTLS. Può anche essere utilizzato anche EAP-PEAP e la configurazione che segue rimane inalterata.

eap.conf:

```
#
#  Whatever you do, do NOT set 'Auth-Type := EAP'.  The server
#  is smart enough to figure this out on its own.  The most
#  common side effect of setting 'Auth-Type := EAP' is that the
#  users then cannot use ANY other authentication method.
#
#       $Id: eap.conf,v 1.4 2004/04/15 18:34:41 aland Exp $
#
        eap {
                #  Invoke the default supported EAP type when
                #  EAP-Identity response is received.
                #
                #  The incoming EAP messages DO NOT specify which EAP
                #  type they will be using, so it MUST be set here.
                #
                #  For now, only one default EAP type may be used at a time.
                #
                #  If the EAP-Type attribute is set by another module,
                #  then that EAP type takes precedence over the
                #  default type configured here.
                #
                default_eap_type = ttls

                #  A list is maintained to correlate EAP-Response
                #  packets with EAP-Request packets.  After a
                #  configurable length of time, entries in the list
                #  expire, and are deleted.
                #
                timer_expire     = 60

                #  There are many EAP types, but the server has support
                #  for only a limited subset.  If the server receives
                #  a request for an EAP type it does not support, then
                #  it normally rejects the request.  By setting this
                #  configuration to "yes", you can tell the server to
                #  instead keep processing the request.  Another module
                #  MUST then be configured to proxy the request to
                #  another RADIUS server which supports that EAP type.
                #
                #  If another module is NOT configured to handle the
                #  request, then the request will still end up being
                #  rejected.
                ignore_unknown_eap_types = no

                # Cisco AP1230B firmware 12.2(13)JA1 has a bug.  When given
                # a User-Name attribute in an Access-Accept, it copies one
                # more byte than it should.
                #
                # We can work around it by configurably adding an extra
                # zero byte.
                cisco_accounting_username_bug = no
```

```
# Supported EAP-types

#
#  We do NOT recommend using EAP-MD5 authentication
#  for wireless connections.  It is insecure, and does
#  not provide for dynamic WEP keys.
#
md5 {
}

# Cisco LEAP
#
#  We do not recommend using LEAP in new deployments.  See:
#  http://www.securiteam.com/tools/5TP012ACKE.html
#
#  Cisco LEAP uses the MS-CHAP algorithm (but not
#  the MS-CHAP attributes) to perform it's authentication.
#
#  As a result, LEAP *requires* access to the plain-text
#  User-Password, or the NT-Password attributes.
#  'System' authentication is impossible with LEAP.
#
leap {
}

#  Generic Token Card.
#
#  Currently, this is only permitted inside of EAP-TTLS,
#  or EAP-PEAP.  The module "challenges" the user with
#  text, and the response from the user is taken to be
#  the User-Password.
#
#  Proxying the tunneled EAP-GTC session is a bad idea,
#  the users password will go over the wire in plain-text,
#  for anyone to see.
#
gtc {
        #  The default challenge, which many clients
        #  ignore..
        #challenge = "Password: "

        #  The plain-text response which comes back
        #  is put into a User-Password attribute,
        #  and passed to another module for
        #  authentication.  This allows the EAP-GTC
        #  response to be checked against plain-text,
        #  or crypt'd passwords.
        #
        #  If you say "Local" instead of "PAP", then
        #  the module will look for a User-Password
        #  configured for the request, and do the
        #  authentication itself.
        #
        auth_type = PAP
}

## EAP-TLS
#
#  To generate ctest certificates, run the script
#
#       ../scripts/certs.sh
#
#  The documents on http://www.freeradius.org/doc
#  are old, but may be helpful.
#
#  See also:
#
#  http://www.dslreports.com/forum/remark,9286052~mode=flat
#
tls {
        private_key_password = whatever
        private_key_file = ${raddbdir}/certs/radius1.key

        #  If Private key & Certificate are located in
        #  the same file, then private_key_file &
        #  certificate_file must contain the same file
        #  name.
        certificate_file = ${raddbdir}/certs/radius1.pem

        #  Trusted Root CA list
        CA_file = ${raddbdir}/certs/infnca.pem

        dh_file = /dev/urandom
        random_file = /dev/urandom

        #
        #  This can never exceed the size of a RADIUS
        #  packet (4096 bytes), and is preferably half
        #  that, to accomodate other attributes in
        #  RADIUS packet.  On most APs the MAX packet
        #  length is configured between 1500 - 1600
        #  In these cases, fragment size should be
        #  1024 or less.
        #
        fragment_size = 1024

        #  include_length is a flag which is
        #  by default set to yes If set to
        #  yes, Total Length of the message is
        #  included in EVERY packet we send.
        #  If set to no, Total Length of the
        #  message is included ONLY in the
        #  First packet of a fragment series.
        #
#       include_length = yes

        #  Check the Certificate Revocation List
        #
        #  1) Copy CA certificates and CRLs to same directory.
        #  2) Execute 'c_rehash <CA certs&CRLs Directory>'.
        #    'c_rehash' is OpenSSL's command.
        #  3) Add 'CA_path=<CA certs&CRLs directory>'
        #      to radiusd.conf's tls section.
        #  4) uncomment the line below.
        #  5) Restart radiusd
#       check_crl = yes

         #
         #  If check_cert_cn is set, the value will
         #  be xlat'ed and checked against the CN
         #  in the client certificate.  If the values
         #  do not match, the certificate verification
         #  will fail rejecting the user.
         #
        check_cert_cn = %{User-Name}
}

#  The TTLS module implements the EAP-TTLS protocol,
#  which can be described as EAP inside of Diameter,
#  inside of TLS, inside of EAP, inside of RADIUS...
#
#  Surprisingly, it works quite well.
#
#  The TTLS module needs the TLS module to be installed
#  and configured, in order to use the TLS tunnel
#  inside of the EAP packet.  You will still need to
#  configure the TLS module, even if you do not want
#  to deploy EAP-TLS in your network.  Users will not
#  be able to request EAP-TLS, as it requires them to
#  have a client certificate.  EAP-TTLS does not
#  require a client certificate.
```

```
#
ttls {
        #  The tunneled EAP session needs a default
        #  EAP type which is separate from the one for
        #  the non-tunneled EAP module.  Inside of the
        #  TTLS tunnel, we recommend using EAP-MD5.
        #  If the request does not contain an EAP
        #  conversation, then this configuration entry
        #  is ignored.
        #default_eap_type = md5

        #  The tunneled authentication request does
        #  not usually contain useful attributes
        #  like 'Calling-Station-Id', etc.  These
        #  attributes are outside of the tunnel,
        #  and normally unavailable to the tunneled
        #  authentication request.
        #
        #  By setting this configuration entry to
        #  'yes', any attribute which NOT in the
        #  tunneled authentication request, but
        #  which IS available outside of the tunnel,
        #  is copied to the tunneled request.
        #
        # allowed values: {no, yes}
        copy_request_to_tunnel = no

        #  The reply attributes sent to the NAS are
          #  usually based on the name of the user
        #  'outside' of the tunnel (usually
        #  'anonymous').  If you want to send the
        #  reply attributes based on the user name
        #  inside of the tunnel, then set this
        #  configuration entry to 'yes', and the reply
        #  to the NAS will be taken from the reply to
        #  the tunneled request.
        #
        # allowed values: {no, yes}
        use_tunneled_reply = no

}

#
#  The tunneled EAP session needs a default EAP type
#  which is separate from the one for the non-tunneled
#  EAP module.  Inside of the TLS/PEAP tunnel, we
#  recommend using EAP-MS-CHAPv2.
#
#  The PEAP module needs the TLS module to be installed
#  and configured, in order to use the TLS tunnel
#  inside of the EAP packet.  You will still need to
#  configure the TLS module, even if you do not want
#  to deploy EAP-TLS in your network.  Users will not
#  be able to request EAP-TLS, as it requires them to
#  have a client certificate.  EAP-PEAP does not
#  require a client certificate.
#
peap {
        #  The tunneled EAP session needs a default
        #  EAP type which is separate from the one for
        #  the non-tunneled EAP module.  Inside of the
        #  PEAP tunnel, we recommend using MS-CHAPv2,
        #  as that is the default type supported by
        #  Windows clients.
        default_eap_type = mschapv2
}

#
#  This takes no configuration.
#
#  Note that it is the EAP MS-CHAPv2 sub-module, not
```

```
               #  the main 'mschap' module.
               #
               #  Note also that in order for this sub-module to work,
               #  the main 'mschap' module MUST ALSO be configured.
               #
               #  This module is the *Microsoft* implementation of MS-CHAPv2
               #  in EAP.  There is another (incompatible) implementation
               #  of MS-CHAPv2 in EAP by Cisco, which FreeRADIUS does not
               #  currently support.
               #
               mschapv2 {
               }
        }
```

Rimane inalterato anche il file *proxy.conf*. Nel caso dell'autenticazione Kerberos bisogna ricordarsi diincludere il Realm all'interno dei domini definiti nel file:

proxy.conf:

```
#
# proxy.conf - proxy radius and realm configuration directives
#
# This file is included by default.  To disable it, you will need
# to modify the PROXY CONFIGURATION section of "radiusd.conf".
#
######################################################################
#
#  Proxy server configuration
#
#  This entry controls the servers behaviour towards ALL other servers
#  to which it sends proxy requests.
#
proxy server {

#
#  If the NAS re-sends the request to us, we can immediately re-send
#  the proxy request to the end server.  To do so, use 'yes' here.
#
#  If this is set to 'no', then we send the retries on our own schedule,
#  and ignore any duplicate NAS requests.
#
#  If you want to have the server send proxy retries ONLY when the NAS
#  sends it's retries to the server, then set this to 'yes', and
#  set the other proxy configuration parameters to 0 (zero).
#
#  Additionally, if you want 'failover' to work, the server must manage
#  retries and timeouts.  Therefore, if this is set to yes, then no
#  failover functionality is possible.
#
        synchronous = no

#
#  The time (in seconds) to wait for a response from the proxy, before
#  re-sending the proxied request.
#
#  If this time is set too high, then the NAS may re-send the request,
#  or it may give up entirely, and reject the user.
#
#  If it is set too low, then the RADIUS server which receives the proxy
#  request will get kicked unnecessarily.
#
        retry_delay = 5

#
#  The number of retries to send before giving up, and sending a reject
#  message to the NAS.
#
```

```
        retry_count = 3

#
#  If the home server does not respond to any of the multiple retries,
#  then FreeRADIUS will stop sending it proxy requests, and mark it 'dead'.
#
#  If there are multiple entries configured for this realm, then the
#  server will fail-over to the next one listed.  If no more are listed,
#  then no requests will be proxied to that realm.
#
#
#  After a configurable 'dead_time', in seconds, FreeRADIUS will
#  speculatively mark the home server active, and start sending requests
#  to it again.
#
#  If this dead time is set too low, then you will lose requests,
#  as FreeRADIUS will quickly switch back to the home server, even if
#  it isn't up again.
#
#  If this dead time is set too high, then FreeRADIUS may take too long
#  to switch back to the primary home server.
#
#  Realistic values for this number are in the range of minutes to hours.
#  (60 to 3600)
#
        dead_time = 120

#  An ldflag attribute for all realms to be included in a round-robin
#  setup must be specified, and that ldflag must be the same for all
#  realms of the same name.
#  Currently (0 or fail_over) and (1 or round_robin) are the
#  supported values for ldflag.  Fail over is the default setup.
#
#  DO NOT INCLUDE LOCAL AUTH/ACCT HOST REALMS IN A ROUND-ROBIN QUEUE.


#
#  If all exact matching realms did not respond, we can try the
#  DEFAULT realm, too.  This is what the server normally does.
#
#  This behaviour may be undesired for some cases.  e.g. You are proxying
#  for two different ISP's, and then act as a general dial-up for Gric.
#  If one of the first two ISP's has their RADIUS server go down, you do
#  NOT want to proxy those requests to GRIC.  Instead, you probably want
#  to just drop the requests on the floor.  In that case, set this value
#  to 'no'.
#
#  allowed values: {yes, no}
#
        default_fallback = yes

#
#  Older versions of the server would pass proxy requests through the
#  'authorize' sections twice; once when the packet was received
#  from the NAS, and again after the reply was received from the home
#  server.  Now that we have a 'post_proxy' section, the replies from
#  the home server should be sent through that, instead of through
#  the 'authorize' section again.
#
#  However, for backwards compatibility, this behaviour is configurable.
#  The default configuration is 'yes', for backwards compatibility.
#  To use ONLY the new 'post_proxy' section, set this value to 'no'.
#
#  allowed values: {yes, no}
#
        post_proxy_authorize = yes

}

#######################################################################
#
```

```
#  Configuration for the proxy realms.
#
#  The information given here is used in conjunction with the 'realms'
#  file.  This format is preferred, as it is more flexible.  The realms
#  listed here take priority over those listed in the 'realms' file.

#  A standard realm entry. A request from "user@company.com" will be
#  sent to radius.company.com as "user", unless the 'nostrip'
#  configuration item is specified.  If the 'nostrip' configuration
#  item is specified, then the request will be proxied as
#  "user@company.com"
#
#realm company.com {
#       type            = radius
#       authhost        = radius.company.com:1600
#       accthost        = radius.company.com:1601
#       secret          = testing123
#}

#  A realm entry with an optional fail-over realm.  A request from
#  "user@isp2.com" will be sent to radius.isp2.com as "user@isp2.com",
#  because the 'nostrip' directive is specified for this realm.
#
#realm isp2.com {
#       type        = radius
#       authhost    = radius.isp2.com:1645
#       accthost    = radius.isp2.com:1646
#       secret      = TheirKey
#       nostrip
#}
#
#  The fail-over realm for isp2.com
#
#realm isp2.com {
#       type        = radius
#       authhost    = radius2.isp2.com:1645
#       accthost    = radius2.isp2.com:1646
#       secret      = TheirKey2
#       nostrip
#}

#
#  1st node serv.com...set up for round-robin.
#
#  The load balancing 'ldflag' attribute can be used to perform
#  load balancing.  Allowed values are 'fail_over' and 'round_robin'.
#
#  If there is no ldflag attribute, or it is set to 'fail_over', then
#  the realms are treated as "fail-over".  That is, the first matching
#  realm is used, unless it is down, in which case the realm "fails
#  over" to the second matching realm.  The process continues until an
#  active matching realm is found, OR the DEFAULT realm is returned.
#
#  If the ldflag attribute is set to 'round_robin', then all active
#  realms of the same name are put into a pool internally in the
#  server, and the proxied requests are evenly divided among the
#  realms in the pool.  For this to work, all realms of the same name
#  MUST have the same value of their 'ldflag' attributes.  Mixing up
#  different types of load balancing schemes for the same realm will
#  cause problems.
#
#  The round_robin load balancing method is a probabilistic method
#  which evenly scatters the requests among the home servers.
#
#  Note that you CANNOT include local auth/acct host realms in a
#  round-robin queue.  Having a server load balance requests to itself
#  doesn't make any sense, as it only doubles the amount of work
#  which is needed to be done.
#
#realm serv.com {
#       type        = radius
#       authhost    = radius.serv.com:1645
#       accthost    = radius.serv.com:1646
#       secret      = TheirKey
#       ldflag      = round_robin
#       nostrip
#}

#
#  Another node for serv.com
#
#realm serv.com {
#       type        = radius
#       authhost    = radius2.serv.com:1645
#       accthost    = radius2.serv.com:1646
#       secret      = TheirKey2
#       ldflag     = round_robin
#       nostrip
#}

#
#  A third round-robin node realm for serv.com
#
#realm serv.com {
#       type        = radius
#       authhost    = radius3.serv.com:1645
#       accthost    = radius3.serv.com:1646
#       secret      = TheirKey2
#       ldflag      = round_robin
#       nostrip
#}
#
#

#
#  This is a local realm.  The requests are NOT proxied,
#  but instead are authenticated by the RADIUS server itself.
#
#  You don't need a secret if BOTH 'authhost' and 'accthost' are
#  set to LOCAL.
#
#realm bla.com {
#       type           = radius
#       authhost       = LOCAL
#       accthost       = LOCAL
#}

#
#  This is a sample entry for iPass.
#
#realm IPASS {
#       type           = radius
#       authhost       = ipass.server.hostname:11812
#       accthost       = ipass.server.hostname:11813
#
        #  The shared secret here must be the same
        #  value as the secret of the NetServer found in the
        #  /usr/ipass/raddb/clients file of your NetServer software.
#       secret         = mysecret
#       nostrip
#}

#
#  This realm is used mainly to cancel proxying.  You can have
#  the "realm suffix" module configured to proxy all requests for
#  a realm, and then later cancel the proxying, based on other
#  configuration.
#
#  For example, you want to terminate PEAP or EAP-TTLS locally,
#  you can add the following to the "users" file:
#
#  DEFAULT EAP-Type == PEAP, Proxy-To-Realm := LOCAL
```

```
#
realm LOCAL {
       type           = radius
       authhost       = LOCAL
       accthost       = LOCAL
}

#
#  This realm is for requests which don't have an explicit realm
#  prefix or suffix.  User names like "bob" will match this one.
#
#realm NULL {
#      type           = radius
#      authhost       = radius.company.com:1600
#      accthost       = radius.company.com:1601
#      secret         = testing123
#}

#
#  This realm is for ALL OTHER requests.
#
#realm DEFAULT {
#      type           = radius
#      authhost       = radius.company.com:1600
#      accthost       = radius.company.com:1601
#      secret         = testing123
#}
realm realm-or-domain.infn.it {
        type            = radius
        authhost        = LOCAL
        accthost        = LOCAL
}

realm DEFAULT {
        type            = radius
        authhost        = radius.garr.net:1812
        accthost        = radius.garr.net:1813
        secret          = ********
        nostrip
}
```

Configurazione di freeradius autenticazione kerberos – autorizzazione ldap

Questa configurazione prevede che sia esistente un'infrastruttura Kerberos per l'autenticazione e LDAP per l'autorizzazione. La piattaforma su cui si è testato questo scenario è Scientific Linux 4.1

È necessario avere istallati i pacchetti relativi a Kerberos:

```
pam_krb5-2.1.8-1
krb5-libs-1.3.4-17
krb5-workstation-1.3.4-17
krb5-devel-1.3.4-17
krb5-auth-dialog-0.2-1
```

Bisogna poi popolare il file `/etc/krb5.keytab` e configurare `/etc/krb5.conf` con il nome del realm e il server kerberos.

Di seguito sono riportati i file di configurazione di freeradius relativi allo scenario in esame.

radiusd.conf:

```
##
## radiusd.conf        -- FreeRADIUS server configuration file.
##
##      http://www.freeradius.org/
##      $Id: radiusd.conf.in,v 1.188 2004/05/13 20:10:19 pnixon Exp $
##

#       The location of other config files and
#       logfiles are declared in this file
#
#       Also general configuration for modules can be done
#       in this file, it is exported through the API to
#       modules that ask for it.
#
#       The configuration variables defined here are of the form ${foo}
#       They are local to this file, and do not change from request to
#       request.
#
#       The per-request variables are of the form %{Attribute-Name}, and
#       are taken from the values of the attribute in the incoming
#       request.  See 'doc/variables.txt' for more information.

prefix = /usr
exec_prefix = /usr
sysconfdir = /etc
localstatedir = /var
sbindir = /usr/sbin
logdir = ${localstatedir}/log/radius
raddbdir = ${sysconfdir}/raddb
radacctdir = ${logdir}/radacct

#  Location of config and logfiles.
confdir = ${raddbdir}
run_dir = ${localstatedir}/run/radiusd

#
#  The logging messages for the server are appended to the
#  tail of this file.
#
log_file = ${logdir}/radius.log

#
# libdir: Where to find the rlm_* modules.
#
#   This should be automatically set at configuration time.
#
#   If the server builds and installs, but fails at execution time
#   with an 'undefined symbol' error, then you can use the libdir
#   directive to work around the problem.
#
#   The cause is usually that a library has been installed on your
#   system in a place where the dynamic linker CANNOT find it.  When
#   executing as root (or another user), your personal environment MAY
#   be set up to allow the dynamic linker to find the library.  When
#   executing as a daemon, FreeRADIUS MAY NOT have the same
#   personalized configuration.
#
#   To work around the problem, find out which library contains that symbol,
#   and add the directory containing that library to the end of 'libdir',
#   with a colon separating the directory names.  NO spaces are allowed.
#
#   e.g. libdir = /usr/local/lib:/opt/package/lib
#
#   You can also try setting the LD_LIBRARY_PATH environment variable
#   in a script which starts the server.
```

```
#
#   If that does not work, then you can re-configure and re-build the
#   server to NOT use shared libraries, via:
#
#       ./configure --disable-shared
#       make
#       make install
#
libdir = /usr/lib

#  pidfile: Where to place the PID of the RADIUS server.
#
#  The server may be signalled while it's running by using this
#  file.
#
#  This file is written when ONLY running in daemon mode.
#
#  e.g.:  kill -HUP `cat /var/run/radiusd/radiusd.pid`
#
pidfile = ${run_dir}/radiusd.pid


# user/group: The name (or #number) of the user/group to run radiusd as.
#
#   If these are commented out, the server will run as the user/group
#   that started it.  In order to change to a different user/group, you
#   MUST be root ( or have root privleges ) to start the server.
#
#   We STRONGLY recommend that you run the server with as few permissions
#   as possible.  That is, if you're not using shadow passwords, the
#   user and group items below should be set to 'nobody'.
#
#    On SCO (ODT 3) use "user = nouser" and "group = nogroup".
#
#  NOTE that some kernels refuse to setgid(group) when the value of
#  (unsigned)group is above 60000; don't use group nobody on these systems!
#
#  On systems with shadow passwords, you might have to set 'group = shadow'
#  for the server to be able to read the shadow password file.  If you can
#  authenticate users while in debug mode, but not in daemon mode, it may be
#  that the debugging mode server is running as a user that can read the
#  shadow info, and the user listed below can not.
#
user = radiusd
group = radiusd

#  max_request_time: The maximum time (in seconds) to handle a request.
#
#  Requests which take more time than this to process may be killed, and
#  a REJECT message is returned.
#
#  WARNING: If you notice that requests take a long time to be handled,
#  then this MAY INDICATE a bug in the server, in one of the modules
#  used to handle a request, OR in your local configuration.
#
#  This problem is most often seen when using an SQL database.  If it takes
#  more than a second or two to receive an answer from the SQL database,
#  then it probably means that you haven't indexed the database.  See your
#  SQL server documentation for more information.
#
#  Useful range of values: 5 to 120
#
max_request_time = 30

#  delete_blocked_requests: If the request takes MORE THAN 'max_request_time'
#  to be handled, then maybe the server should delete it.
#
#  If you're running in threaded, or thread pool mode, this setting
#  should probably be 'no'.  Setting it to 'yes' when using a threaded
#  server MAY cause the server to crash!
#
```

```
delete_blocked_requests = no

#  cleanup_delay: The time to wait (in seconds) before cleaning up
#  a reply which was sent to the NAS.
#
#  The RADIUS request is normally cached internally for a short period
#  of time, after the reply is sent to the NAS.  The reply packet may be
#  lost in the network, and the NAS will not see it.  The NAS will then
#  re-send the request, and the server will respond quickly with the
#  cached reply.
#
#  If this value is set too low, then duplicate requests from the NAS
#  MAY NOT be detected, and will instead be handled as seperate requests.
#
#  If this value is set too high, then the server will cache too many
#  requests, and some new requests may get blocked.  (See 'max_requests'.)
#
#  Useful range of values: 2 to 10
#
cleanup_delay = 5

#  max_requests: The maximum number of requests which the server keeps
#  track of.  This should be 256 multiplied by the number of clients.
#  e.g. With 4 clients, this number should be 1024.
#
#  If this number is too low, then when the server becomes busy,
#  it will not respond to any new requests, until the 'cleanup_delay'
#  time has passed, and it has removed the old requests.
#
#  If this number is set too high, then the server will use a bit more
#  memory for no real benefit.
#
#  If you aren't sure what it should be set to, it's better to set it
#  too high than too low.  Setting it to 1000 per client is probably
#  the highest it should be.
#
#  Useful range of values: 256 to infinity
#
max_requests = 1024

#  bind_address:  Make the server listen on a particular IP address, and
#  send replies out from that address.  This directive is most useful
#  for machines with multiple IP addresses on one interface.
#
#  It can either contain "*", or an IP address, or a fully qualified
#  Internet domain name.  The default is "*"
#
#  As of 1.0, you can also use the "listen" directive.  See below for
#  more information.
#
bind_address = *

#  port: Allows you to bind FreeRADIUS to a specific port.
#
#  The default port that most NAS boxes use is 1645, which is historical.
#  RFC 2138 defines 1812 to be the new port.  Many new servers and
#  NAS boxes use 1812, which can create interoperability problems.
#
#  The port is defined here to be 0 so that the server will pick up
#  the machine's local configuration for the radius port, as defined
#  in /etc/services.
#
#  If you want to use the default RADIUS port as defined on your server,
#  (usually through 'grep radius /etc/services') set this to 0 (zero).
#
#  A port given on the command-line via '-p' over-rides this one.
#
#  As of 1.0, you can also use the "listen" directive.  See below for
#  more information.
#
port = 0
```

```
#
#  By default, the server uses "bind_address" to listen to all IP's
#  on a machine, or just one IP.  The "port" configuration is used
#  to select the authentication port used when listening on those
#  addresses.
#
#  If you want the server to listen on additional addresses, you can
#  use the "listen" section.  A sample section (commented out) is included
#  below.  This "listen" section duplicates the functionality of the
#  "bind_address" and "port" configuration entries, but it only listens
#  for authentication packets.
#
#  If you comment out the "bind_address" and "port" configuration entries,
#  then it becomes possible to make the server accept only accounting,
#  or authentication packets.  Previously, it always listened for both
#  types of packets, and it was impossible to make it listen for only
#  one type of packet.
#
#listen {
        #  IP address on which to listen.
        #  Allowed values are:
        #       dotted quad (1.2.3.4)
        #       hostname    (radius.example.com)
        #       wildcard    (*)
#       ipaddr = *

        #  Port on which to listen.
        #  Allowed values are:
        #       integer port number (1812)
        #       0 means "use /etc/services for the proper port"
#       port = 0

        #  Type of packets to listen for.
        #  Allowed values are:
        #       auth    listen for authentication packets
        #       acct    listen for accounting packets
        #
#       type = auth
#}


#  hostname_lookups: Log the names of clients or just their IP addresses
#  e.g., www.freeradius.org (on) or 206.47.27.232 (off).
#
#  The default is 'off' because it would be overall better for the net
#  if people had to knowingly turn this feature on, since enabling it
#  means that each client request will result in AT LEAST one lookup
#  request to the nameserver.   Enabling hostname_lookups will also
#  mean that your server may stop randomly for 30 seconds from time
#  to time, if the DNS requests take too long.
#
#  Turning hostname lookups off also means that the server won't block
#  for 30 seconds, if it sees an IP address which has no name associated
#  with it.
#
#  allowed values: {no, yes}
#
hostname_lookups = no

#  Core dumps are a bad thing.  This should only be set to 'yes'
#  if you're debugging a problem with the server.
#
#  allowed values: {no, yes}
#
allow_core_dumps = no

#  Regular expressions
#
#  These items are set at configure time.  If they're set to "yes",
#  then setting them to "no" turns off regular expression support.
```

```
#
#  If they're set to "no" at configure time, then setting them to "yes"
#  WILL NOT WORK.  It will give you an error.
#
regular_expressions    = yes
extended_expressions   = yes

#  Log the full User-Name attribute, as it was found in the request.
#
# allowed values: {no, yes}
#
log_stripped_names = no

#  Log authentication requests to the log file.
#
#  allowed values: {no, yes}
#
log_auth = yes

#  Log passwords with the authentication requests.
#  log_auth_badpass  - logs password if it's rejected
#  log_auth_goodpass - logs password if it's correct
#
#  allowed values: {no, yes}
#
log_auth_badpass = no
log_auth_goodpass = no

# usercollide:  Turn "username collision" code on and off.  See the
# "doc/duplicate-users" file
#
#  WARNING
#  !!!!!!!  Setting this to "yes" may result in the server behaving
#  !!!!!!!  strangely.  The "username collision" code will ONLY work
#  !!!!!!!  with clear-text passwords.  Even then, it may not do what
#  !!!!!!!  you want, or what you expect.
#  !!!!!!!
#  !!!!!!!  We STRONGLY RECOMMEND that you do not use this feature,
#  !!!!!!!  and that you find another way of acheiving the same goal.
#  !!!!!!!
#  !!!!!!!  e,g. module fail-over.  See 'doc/configurable_failover'
#  WARNING
#
usercollide = no

# lower_user / lower_pass:
# Lower case the username/password "before" or "after"
# attempting to authenticate.
#
#  If "before", the server will first modify the request and then try
#  to auth the user.  If "after", the server will first auth using the
#  values provided by the user.  If that fails it will reprocess the
#  request after modifying it as you specify below.
#
#  This is as close as we can get to case insensitivity.  It is the
#  admin's job to ensure that the username on the auth db side is
#  *also* lowercase to make this work
#
# Default is 'no' (don't lowercase values)
# Valid values = "before" / "after" / "no"
#
lower_user = no
lower_pass = no

# nospace_user / nospace_pass:
#
#  Some users like to enter spaces in their username or password
#  incorrectly.  To save yourself the tech support call, you can
#  eliminate those spaces here:
#
# Default is 'no' (don't remove spaces)
# Valid values = "before" / "after" / "no" (explanation above)
#
nospace_user = no
nospace_pass = no

#  The program to execute to do concurrency checks.
checkrad = ${sbindir}/checkrad

# SECURITY CONFIGURATION
#
#  There may be multiple methods of attacking on the server.  This
#  section holds the configuration items which minimize the impact
#  of those attacks
#
security {
        #
        #  max_attributes: The maximum number of attributes
        #  permitted in a RADIUS packet.  Packets which have MORE
        #  than this number of attributes in them will be dropped.
        #
        #  If this number is set too low, then no RADIUS packets
        #  will be accepted.
        #
        #  If this number is set too high, then an attacker may be
        #  able to send a small number of packets which will cause
        #  the server to use all available memory on the machine.
        #
        #  Setting this number to 0 means "allow any number of attributes"
        max_attributes = 200

        #
        #  delayed_reject: When sending an Access-Reject, it can be
        #  delayed for a few seconds.  This may help slow down a DoS
        #  attack.  It also helps to slow down people trying to brute-force
        #  crack a users password.
        #
        #  Setting this number to 0 means "send rejects immediately"
        #
        #  If this number is set higher than 'cleanup_delay', then the
        #  rejects will be sent at 'cleanup_delay' time, when the request
        #  is deleted from the internal cache of requests.
        #
        #  Useful ranges: 1 to 5
        reject_delay = 1

        #
        #  status_server: Whether or not the server will respond
        #  to Status-Server requests.
        #
        #  Normally this should be set to "no", because they're useless.
        #  See: http://www.freeradius.org/rfc/rfc2865.html#Keep-Alives
        #
        #  However, certain NAS boxes may require them.
        #
        #  When sent a Status-Server message, the server responds with
        #  an Access-Accept packet, containing a Reply-Message attribute,
        #  which is a string describing how long the server has been
        #  running.
        #
        status_server = no
}

# PROXY CONFIGURATION
#
#  proxy_requests: Turns proxying of RADIUS requests on or off.
#
#  The server has proxying turned on by default.  If your system is NOT
#  set up to proxy requests to another server, then you can turn proxying
#  off here.  This will save a small amount of resources on the server.
#
#  If you have proxying turned off, and your configuration files say
#  to proxy a request, then an error message will be logged.
#
#  To disable proxying, change the "yes" to "no", and comment the
#  $INCLUDE line.
#
#  allowed values: {no, yes}
#
proxy_requests  = yes
$INCLUDE  ${confdir}/proxy.conf


# CLIENTS CONFIGURATION
#
#  Client configuration is defined in "clients.conf".
#

#  The 'clients.conf' file contains all of the information from the old
#  'clients' and 'naslist' configuration files.  We recommend that you
#  do NOT use 'client's or 'naslist', although they are still
#  supported.
#
#  Anything listed in 'clients.conf' will take precedence over the
#  information from the old-style configuration files.
#
$INCLUDE  ${confdir}/clients.conf


# SNMP CONFIGURATION
#
#  Snmp configuration is only valid if SNMP support was enabled
#  at compile time.
#
#  To enable SNMP querying of the server, set the value of the
#  'snmp' attribute to 'yes'
#
snmp    = no
$INCLUDE  ${confdir}/snmp.conf


# THREAD POOL CONFIGURATION
#
#  The thread pool is a long-lived group of threads which
#  take turns (round-robin) handling any incoming requests.
#
#  You probably want to have a few spare threads around,
#  so that high-load situations can be handled immediately.  If you
#  don't have any spare threads, then the request handling will
#  be delayed while a new thread is created, and added to the pool.
#
#  You probably don't want too many spare threads around,
#  otherwise they'll be sitting there taking up resources, and
#  not doing anything productive.
#
#  The numbers given below should be adequate for most situations.
#
thread pool {
        #  Number of servers to start initially --- should be a reasonable
        #  ballpark figure.
        start_servers = 5

        #  Limit on the total number of servers running.
        #
        #  If this limit is ever reached, clients will be LOCKED OUT, so it
        #  should NOT BE SET TOO LOW.  It is intended mainly as a brake to
        #  keep a runaway server from taking the system with it as it spirals
        #  down...
        #
        #  You may find that the server is regularly reaching the
        #  'max_servers' number of threads, and that increasing
        #  'max_servers' doesn't seem to make much difference.
        #
        #  If this is the case, then the problem is MOST LIKELY that
        #  your back-end databases are taking too long to respond, and
        #  are preventing the server from responding in a timely manner.
        #
        #  The solution is NOT do keep increasing the 'max_servers'
        #  value, but instead to fix the underlying cause of the
        #  problem: slow database, or 'hostname_lookups=yes'.
        #
        #  For more information, see 'max_request_time', above.
        #
        max_servers = 32

        #  Server-pool size regulation.  Rather than making you guess
        #  how many servers you need, FreeRADIUS dynamically adapts to
        #  the load it sees, that is, it tries to maintain enough
        #  servers to handle the current load, plus a few spare
        #  servers to handle transient load spikes.
        #
        #  It does this by periodically checking how many servers are
        #  waiting for a request.  If there are fewer than
        #  min_spare_servers, it creates a new spare.  If there are
        #  more than max_spare_servers, some of the spares die off.
        #  The default values are probably OK for most sites.
        #
        min_spare_servers = 3
        max_spare_servers = 10

        #  There may be memory leaks or resource allocation problems with
        #  the server.  If so, set this value to 300 or so, so that the
        #  resources will be cleaned up periodically.
        #
        #  This should only be necessary if there are serious bugs in the
        #  server which have not yet been fixed.
        #
        #  '0' is a special value meaning 'infinity', or 'the servers never
        #  exit'
        max_requests_per_server = 0
}

# MODULE CONFIGURATION
#
#  The names and configuration of each module is located in this section.
#
#  After the modules are defined here, they may be referred to by name,
#  in other sections of this configuration file.
#
modules {
        #
        #  Each module has a configuration as follows:
        #
        #       name [ instance ] {
        #               config_item = value
        #               ...
        #       }
        #
        #  The 'name' is used to load the 'rlm_name' library
        #  which implements the functionality of the module.
        #
        #  The 'instance' is optional.  To have two different instances
        #  of a module, it first must be referred to by 'name'.
        #  The different copies of the module are then created by
        #  inventing two 'instance' names, e.g. 'instance1' and 'instance2'
        #
        #  The instance names can then be used in later configuration
        #  INSTEAD of the original 'name'.  See the 'radutmp' configuration
        #  below for an example.
        #

        # PAP module to authenticate users based on their stored password
        #
        #  Supports multiple encryption schemes
        #  clear: Clear text
        #  crypt: Unix crypt
        #    md5: MD5 ecnryption
        #   sha1: SHA1 encryption.
        #  DEFAULT: crypt
        pap {
                encryption_scheme = crypt
        }

        # CHAP module
        #
        #  To authenticate requests containing a CHAP-Password attribute.
        #
        chap {
                authtype = CHAP
        }

        # Pluggable Authentication Modules
        #
        #  For Linux, see:
        #       http://www.kernel.org/pub/linux/libs/pam/index.html
        #
        #  WARNING: On many systems, the system PAM libraries have
        #           memory leaks!  We STRONGLY SUGGEST that you do not
        #           use PAM for authentication, due to those memory leaks.
        #
        pam {
                #
                #  The name to use for PAM authentication.
                #  PAM looks in /etc/pam.d/${pam_auth_name}
                #  for it's configuration.  See 'redhat/radiusd-pam'
                #  for a sample PAM configuration file.
                #
                #  Note that any Pam-Auth attribute set in the 'authorize'
                #  section will over-ride this one.
                #
                pam_auth = radiusd
        }
        # Kerberos
        krb5 {
                keytab = /etc/krb5.keytab
                #service_principal =
        }

        # Unix /etc/passwd style authentication
        #
        unix {
                #
                #  Cache /etc/passwd, /etc/shadow, and /etc/group
                #
                #  The default is to NOT cache them.
                #
                #  For FreeBSD and NetBSD, you do NOT want to enable
                #  the cache, as it's password lookups are done via a
                #  database, so set this value to 'no'.
                #
                #  Some systems (e.g. RedHat Linux with pam_pwbd) can
                #  take *seconds* to check a password, when th passwd
                #  file containing 1000's of entries.  For those systems,
                #  you should set the cache value to 'yes', and set
                #  the locations of the 'passwd', 'shadow', and 'group'
                #  files, below.
                #
                # allowed values: {no, yes}
                cache = no

                # Reload the cache every 600 seconds (10mins). 0 to disable.
                cache_reload = 600

                #
                #  Define the locations of the normal passwd, shadow, and
```

```
		#  group files.
		#
		#  'shadow' is commented out by default, because not all
		#  systems have shadow passwords.
		#
		#  To force the module to use the system password functions,
		#  instead of reading the files, leave the following entries
		#  commented out.
		#
		#  This is required for some systems, like FreeBSD,
		#  and Mac OSX.
		#
		#	passwd = /etc/passwd
		shadow = /etc/shadow
		#	group = /etc/group

		#
		#  The location of the "wtmp" file.
		#  This should be moved to it's own module soon.
		#
		#  The only use for 'radlast'.  If you don't use
		#  'radlast', then you can comment out this item.
		#
		radwtmp = ${logdir}/radwtmp
	}

	#  Extensible Authentication Protocol
	#
	#  For all EAP related authentications.
	#  Now in another file, because it is very large.
	#
$INCLUDE ${confdir}/eap.conf

	# Microsoft CHAP authentication
	#
	#  This module supports MS-CHAP and MS-CHAPv2 authentication.
	#  It also enforces the SMB-Account-Ctrl attribute.
	#
	mschap {
		#
		#  As of 0.9, the mschap module does NOT support
		#  reading from /etc/smbpasswd.
		#
		#  If you are using /etc/smbpasswd, see the 'passwd'
		#  module for an example of how to use /etc/smbpasswd

		# authtype value, if present, will be used
		# to overwrite (or add) Auth-Type during
		# authorization. Normally should be MS-CHAP
		authtype = MS-CHAP

		# if use_mppe is not set to no mschap will
		# add MS-CHAP-MPPE-Keys for MS-CHAPv1 and
		# MS-MPPE-Recv-Key/MS-MPPE-Send-Key for MS-CHAPv2
		#
		#use_mppe = no

		# if mppe is enabled require_encryption makes
		# encryption moderate
		#
		#require_encryption = yes

		# require_strong always requires 128 bit key
		# encryption
		#
		#require_strong = yes

		# Windows sends us a username in the form of
		# DOMAIN\user, but sends the challenge response
		# based on only the user portion.  This hack
		# corrects for that incorrect behavior.
```

```
                #
                #with_ntdomain_hack = no

                # The module can perform authentication itself, OR
                # use a Windows Domain Controller.  This configuration
                # directive tells the module to call the ntlm_auth
                # program, which will do the authentication, and return
                # the NT-Key.  Note that you MUST have "winbindd" and
                # "nmbd" running on the local machine for ntlm_auth
                # to work.  See the ntlm_auth program documentation
                # for details.
                #
                # Be VERY careful when editing the following line!
                #
                #ntlm_auth = "/path/to/ntlm_auth --request-nt-key --username=%{Stripped-
User-Name:-%{User-Name:-None}} --challenge=%{mschap:Challenge:-00} --nt-
response=%{mschap:NT-Response:-00}"
        }

        # Lightweight Directory Access Protocol (LDAP)
        #
        #  This module definition allows you to use LDAP for
        #  authorization and authentication (Auth-Type := LDAP)
        #
        #  See doc/rlm_ldap for description of configuration options
        #  and sample authorize{} and authenticate{} blocks
        ldap {
                server = "domethin.somedomain.infn.it"
                # identity = "cn=admin,o=My Org,c=UA"
                # password = mypass
                basedn = "ou=people,o=something,o=infn,c=it"
                filter = "(uid=%{Stripped-User-Name:-%{User-Name}})"
                # base_filter = "(objectclass=radiusprofile)"

                # set this to 'yes' to use TLS encrypted connections
                # to the LDAP database by using the StartTLS extended
                # operation.
                # The StartTLS operation is supposed to be used with normal
                # ldap connections instead of using ldaps (port 689) connections
                start_tls = no

                # tls_cacertfile      = /path/to/cacert.pem
                # tls_cacertdir               = /path/to/ca/dir/
                # tls_certfile        = /path/to/radius.crt
                # tls_keyfile         = /path/to/radius.key
                # tls_randfile        = /path/to/rnd
                # tls_require_cert    = "demand"

                # default_profile = "cn=radprofile,ou=dialup,o=My Org,c=UA"
                # profile_attribute = "radiusProfileDn"
                #access_attr = "dialupAccess"

                # Mapping of RADIUS dictionary attributes to LDAP
                # directory attributes.
                dictionary_mapping = ${raddbdir}/ldap.attrmap

                ldap_connections_number = 5

                #
                # NOTICE: The password_header directive is NOT case insensitive
                #
                # password_header = "{clear}"
                #
                #  The server can usually figure this out on its own, and pull
                #  the correct User-Password or NT-Password from the database.
                #
                #  Note that NT-Passwords MUST be stored as a 32-digit hex
                #  string, and MUST start off with "0x", such as:
                #
                #       0x000102030405060708090a0b0c0d0e0f
                #
```

```
		#  Without the leading "0x", NT-Passwords will not work.
		#  This goes for NT-Passwords stored in SQL, too.
		#
		# password_attribute = userPassword
		# groupname_attribute = cn
		# groupmembership_filter = "(|(&(objectClass=GroupOfNames)(member=%{Ldap-
UserDn}))(&(objectClass=GroupOfUniqueNames)(uniquemember=%{Ldap-UserDn})))"
		# groupmembership_attribute = radiusGroupName
		timeout = 4
		timelimit = 3
		net_timeout = 1
		# compare_check_items = yes
		# do_xlat = yes
		# access_attr_used_for_allow = yes
	}

	# passwd module allows to do authorization via any passwd-like
	# file and to extract any attributes from these modules
	#
	# parameters are:
	#   filename - path to filename
	#   format - format for filename record. This parameters
	#            correlates record in the passwd file and RADIUS
	#            attributes.
	#
	#            Field marked as '*' is key field. That is, the parameter
	#            with this name from the request is used to search for
	#            the record from passwd file
	#            Attribute marked as '=' is added to reply_itmes instead
	#            of default configure_itmes
	#           Attribute marked as '~' is added to request_items
	#
	#            Field marked as ',' may contain a comma separated list
	#            of attributes.
	#   authtype - if record found this Auth-Type is used to authenticate
	#            user
	#   hashsize - hashtable size. If 0 or not specified records are not
	#            stored in memory and file is red on every request.
	#   allowmultiplekeys - if few records for every key are allowed
	#   ignorenislike - ignore NIS-related records
	#   delimiter - symbol to use as a field separator in passwd file,
	#            for format ':' symbol is always used. '\0', '\n' are
	 #           not allowed
	#

	#  An example configuration for using /etc/smbpasswd.
	#
	#passwd etc_smbpasswd {
	#	filename = /etc/smbpasswd
	#	format = "*User-Name::LM-Password:NT-Password:SMB-Account-CTRL-TEXT::"
	#	authtype = MS-CHAP
	#	hashsize = 100
	#	ignorenislike = no
	#	allowmultiplekeys = no
	#}

	#  Similar configuration, for the /etc/group file. Adds a Group-Name
	#  attribute for every group that the user is member of.
	#
	#passwd etc_group {
	#	filename = /etc/group
	#	format = "=Group-Name:::*,User-Name"
	#	hashsize = 50
	#	ignorenislike = yes
	#	allowmultiplekeys = yes
	#	delimiter = ":"
	#}

	# Realm module, for proxying.
	#
	#  You can have multiple instances of the realm module to
#  support multiple realm syntaxs at the same time.  The
#  search order is defined by the order in the authorize and
#  preacct sections.
#
#  Four config options:
#       format         -  must be 'prefix' or 'suffix'
#       delimiter      -  must be a single character
#       ignore_default -  set to 'yes' or 'no'
#        ignore_null    -  set to 'yes' or 'no'
#
#  ignore_default and ignore_null can be set to 'yes' to prevent
#  the module from matching against DEFAULT or NULL realms.  This
#  may be useful if you have have multiple instances of the
#  realm module.
#
#  They both default to 'no'.
#

#  'realm/username'
#
#  Using this entry, IPASS users have their realm set to "IPASS".
realm IPASS {
       format = prefix
       delimiter = "/"
       ignore_default = no
       ignore_null = no
}

#  'username@realm'
#
realm suffix {
       format = suffix
       delimiter = "@"
       ignore_default = no
       ignore_null = no
}

#  'username%realm'
#
realm realmpercent {
       format = suffix
       delimiter = "%"
       ignore_default = no
       ignore_null = no
}

#
#  'domain\user'
#
realm ntdomain {
       format = prefix
       delimiter = "\\"
       ignore_default = no
       ignore_null = no
}

#  A simple value checking module
#
#  It can be used to check if an attribute value in the request
#  matches a (possibly multi valued) attribute in the check
#  items This can be used for example for caller-id
#  authentication.  For the module to run, both the request
#  attribute and the check items attribute must exist
#
#  i.e.
#  A user has an ldap entry with 2 radiusCallingStationId
#  attributes with values "12345678" and "12345679".  If we
#  enable rlm_checkval, then any request which contains a
#  Calling-Station-Id with one of those two values will be
#  accepted.  Requests with other values for
#  Calling-Station-Id will be rejected.
```

```
        #
        #  Regular expressions in the check attribute value are allowed
        #  as long as the operator is '=~'
        #
        checkval {
                # The attribute to look for in the request
                item-name = Calling-Station-Id

                # The attribute to look for in check items. Can be multi valued
                check-name = Calling-Station-Id

                # The data type. Can be
                # string,integer,ipaddr,date,abinary,octets
                data-type = string

                # If set to yes and we dont find the item-name attribute in the
                # request then we send back a reject
                # DEFAULT is no
                #notfound-reject = no
        }

        #  rewrite arbitrary packets.  Useful in accounting and authorization.
        #
        #
        #  The module can also use the Rewrite-Rule attribute. If it
        #  is set and matches the name of the module instance, then
        #  that module instance will be the only one which runs.
        #
        #  Also if new_attribute is set to yes then a new attribute
        #  will be created containing the value replacewith and it
        #  will be added to searchin (packet, reply, proxy, proxy_reply or config).
        # searchfor,ignore_case and max_matches will be ignored in that case.
        #
        # Backreferences are supported: %{0} will contain the string the whole match
        # and %{1} to %{8} will contain the contents of the 1st to the 8th parentheses
        #
        # If max_matches is greater than one the backreferences will correspond to the
        # first match

        #
        #attr_rewrite sanecallerid {
        #       attribute = Called-Station-Id
                # may be "packet", "reply", "proxy", "proxy_reply" or "config"
        #       searchin = packet
        #       searchfor = "[+ ]"
        #       replacewith = ""
        #       ignore_case = no
        #       new_attribute = no
        #       max_matches = 10
        #       ## If set to yes then the replace string will be appended to the original
string
        #       append = no
        #}

        # Preprocess the incoming RADIUS request, before handing it off
        # to other modules.
        #
        #  This module processes the 'huntgroups' and 'hints' files.
        #  In addition, it re-writes some weird attributes created
        #  by some NASes, and converts the attributes into a form which
        #  is a little more standard.
        #
        preprocess {
                huntgroups = ${confdir}/huntgroups
                hints = ${confdir}/hints

                # This hack changes Ascend's wierd port numberings
                # to standard 0-??? port numbers so that the "+" works
                # for IP address assignments.
                with_ascend_hack = no
                ascend_channels_per_line = 23
                # Windows NT machines often authenticate themselves as
                # NT_DOMAIN\username
                #
                # If this is set to 'yes', then the NT_DOMAIN portion
                # of the user-name is silently discarded.
                #
                # This configuration entry SHOULD NOT be used.
                # See the "realms" module for a better way to handle
                # NT domains.
                with_ntdomain_hack = no

                # Specialix Jetstream 8500 24 port access server.
                #
                # If the user name is 10 characters or longer, a "/"
                # and the excess characters after the 10th are
                # appended to the user name.
                #
                # If you're not running that NAS, you don't need
                # this hack.
                with_specialix_jetstream_hack = no

                # Cisco sends it's VSA attributes with the attribute
                # name *again* in the string, like:
                #
                #   H323-Attribute = "h323-attribute=value".
                #
                # If this configuration item is set to 'yes', then
                # the redundant data in the the attribute text is stripped
                # out.  The result is:
                #
                #  H323-Attribute = "value"
                #
                # If you're not running a Cisco NAS, you don't need
                # this hack.
                with_cisco_vsa_hack = no
        }

        # Livingston-style 'users' file
        #
        files {
                usersfile = ${confdir}/users
                acctusersfile = ${confdir}/acct_users

                #  If you want to use the old Cistron 'users' file
                #  with FreeRADIUS, you should change the next line
                #  to 'compat = cistron'.  You can the copy your 'users'
                #  file from Cistron.
                compat = no
        }

        # Write a detailed log of all accounting records received.
        #
        detail {
                #  Note that we do NOT use NAS-IP-Address here, as
                #  that attribute MAY BE from the originating NAS, and
                #  NOT from the proxy which actually sent us the
                #  request.  The Client-IP-Address attribute is ALWAYS
                #  the address of the client which sent us the
                #  request.
                #
                #  The following line creates a new detail file for
                #  every radius client (by IP address or hostname).
                #  In addition, a new detail file is created every
                #  day, so that the detail file doesn't have to go
                #  through a 'log rotation'
                #
                #  If your detail files are large, you may also want
                #  to add a ':%H' (see doc/variables.txt) to the end
                #  of it, to create a new detail file every hour, e.g.:
                #
```

```
	#   ..../detail-%Y%m%d:%H
	#
	#  This will create a new detail file for every hour.
	#
	detailfile = ${radacctdir}/%{Client-IP-Address}/detail-%Y%m%d

	#
	#  The Unix-style permissions on the 'detail' file.
	#
	#  The detail file often contains secret or private
	#  information about users.  So by keeping the file
	#  permissions restrictive, we can prevent unwanted
	#  people from seeing that information.
	detailperm = 0600
}

#
#  Many people want to log authentication requests.
#  Rather than modifying the server core to print out more
#  messages, we can use a different instance of the 'detail'
#  module, to log the authentication requests to a file.
#
#  You will also need to un-comment the 'auth_log' line
#  in the 'authorize' section, below.
#
# detail auth_log {
	# detailfile = ${radacctdir}/%{Client-IP-Address}/auth-detail-%Y%m%d

	#
	#  This MUST be 0600, otherwise anyone can read
	#  the users passwords!
	# detailperm = 0600
# }

#
#  This module logs authentication reply packets sent
#  to a NAS.  Both Access-Accept and Access-Reject packets
#  are logged.
#
#  You will also need to un-comment the 'reply_log' line
#  in the 'post-auth' section, below.
#
# detail reply_log {
	# detailfile = ${radacctdir}/%{Client-IP-Address}/reply-detail-%Y%m%d

	#
	#  This MUST be 0600, otherwise anyone can read
	#  the users passwords!
	# detailperm = 0600
# }

#
#  This module logs packets proxied to a home server.
#
#  You will also need to un-comment the 'pre_proxy_log' line
#  in the 'pre-proxy' section, below.
#
# detail pre_proxy_log {
	# detailfile = ${radacctdir}/%{Client-IP-Address}/pre-proxy-detail-%Y%m%d

	#
	#  This MUST be 0600, otherwise anyone can read
	#  the users passwords!
	# detailperm = 0600
# }

#
#  This module logs response packets from a home server.
#
#  You will also need to un-comment the 'post_proxy_log' line
#  in the 'post-proxy' section, below.
```

```
	#
	# detail post_proxy_log {
		# detailfile = ${radacctdir}/%{Client-IP-Address}/post-proxy-detail-%Y%m%d

		#
		#  This MUST be 0600, otherwise anyone can read
		#  the users passwords!
		# detailperm = 0600
	# }

	# Create a unique accounting session Id.  Many NASes re-use or
	# repeat values for Acct-Session-Id, causing no end of
	# confusion.
	#
	#  This module will add a (probably) unique session id
	#  to an accounting packet based on the attributes listed
	#  below found in the packet.  See doc/rlm_acct_unique for
	#  more information.
	#
	acct_unique {
		key = "User-Name, Acct-Session-Id, NAS-IP-Address, Client-IP-Address, NAS-
Port"
	}


	#  Include another file that has the SQL-related configuration.
	#  This is another file only because it tends to be big.
	#
	#  The following configuration file is for use with MySQL.
	#
	#  For Postgresql, use:              ${confdir}/postgresql.conf
	#  For MS-SQL, use:           ${confdir}/mssql.conf
	#  For Oracle, use:           ${confdir}/oraclesql.conf
	#
	$INCLUDE  ${confdir}/sql.conf


	#  For Cisco VoIP specific accounting with Postgresql,
	#  use:        ${confdir}/pgsql-voip.conf
	#
	#  You will also need the sql schema from:
	#       src/billing/cisco_h323_db_schema-postgres.sql
	#  Note: This config can be use AS WELL AS the standard sql
	#  config if you need SQL based Auth


	#  Write a 'utmp' style file, of which users are currently
	#  logged in, and where they've logged in from.
	#
	#  This file is used mainly for Simultaneous-Use checking,
	#  and also 'radwho', to see who's currently logged in.
	#
	radutmp {
		#  Where the file is stored.  It's not a log file,
		#  so it doesn't need rotating.
		#
		filename = ${logdir}/radutmp

		#  The field in the packet to key on for the
		#  'user' name,  If you have other fields which you want
		#  to use to key on to control Simultaneous-Use,
		#  then you can use them here.
		#
		#  Note, however, that the size of the field in the
		#  'utmp' data structure is small, around 32
		#  characters, so that will limit the possible choices
		#  of keys.
		#
		#  You may want instead: %{Stripped-User-Name:-%{User-Name}}
		username = %{User-Name}
```

```
        #  Whether or not we want to treat "user" the same
        #  as "USER", or "User".  Some systems have problems
        #  with case sensitivity, so this should be set to
        #  'no' to enable the comparisons of the key attribute
        #  to be case insensitive.
        #
        case_sensitive = yes

        #  Accounting information may be lost, so the user MAY
        #  have logged off of the NAS, but we haven't noticed.
        #  If so, we can verify this information with the NAS,
        #
        #  If we want to believe the 'utmp' file, then this
        #  configuration entry can be set to 'no'.
        #
        check_with_nas = yes

        # Set the file permissions, as the contents of this file
        # are usually private.
        perm = 0600

        callerid = "yes"
}

# "Safe" radutmp - does not contain caller ID, so it can be
# world-readable, and radwho can work for normal users, without
# exposing any information that isn't already exposed by who(1).
#
# This is another 'instance' of the radutmp module, but it is given
# then name "sradutmp" to identify it later in the "accounting"
# section.
radutmp sradutmp {
        filename = ${logdir}/sradutmp
        perm = 0644
        callerid = "no"
}

# attr_filter - filters the attributes received in replies from
# proxied servers, to make sure we send back to our RADIUS client
# only allowed attributes.
attr_filter {
        attrsfile = ${confdir}/attrs
}

#  counter module:
#  This module takes an attribute (count-attribute).
#  It also takes a key, and creates a counter for each unique
#  key.  The count is incremented when accounting packets are
#  received by the server.  The value of the increment depends
#  on the attribute type.
#  If the attribute is Acct-Session-Time or of an integer type we add the
#  value of the attribute. If it is anything else we increase the
#  counter by one.
#
#  The 'reset' parameter defines when the counters are all reset to
#  zero.  It can be hourly, daily, weekly, monthly or never.
#
#  hourly: Reset on 00:00 of every hour
#  daily: Reset on 00:00:00 every day
#  weekly: Reset on 00:00:00 on sunday
#  monthly: Reset on 00:00:00 of the first day of each month
#
#  It can also be user defined. It should be of the form:
#  num[hdwm] where:
#  h: hours, d: days, w: weeks, m: months
#  If the letter is ommited days will be assumed. In example:
#  reset = 10h (reset every 10 hours)
#  reset = 12  (reset every 12 days)
#
#
```

```
#  The check-name attribute defines an attribute which will be
#  registered by the counter module and can be used to set the
#  maximum allowed value for the counter after which the user
#  is rejected.
#  Something like:
#
#  DEFAULT Max-Daily-Session := 36000
#          Fall-Through = 1
#
#  You should add the counter module in the instantiate
#  section so that it registers check-name before the files
#  module reads the users file.
#
#  If check-name is set and the user is to be rejected then we
#  send back a Reply-Message and we log a Failure-Message in
#  the radius.log
#  If the count attribute is Acct-Session-Time then on each login
#  we send back the remaining online time as a Session-Timeout attribute
#
#  The counter-name can also be used instead of using the check-name
#  like below:
#
#  DEFAULT  Daily-Session-Time > 3600, Auth-Type = Reject
#      Reply-Message = "You've used up more than one hour today"
#
#  The allowed-servicetype attribute can be used to only take
#  into account specific sessions. For example if a user first
#  logs in through a login menu and then selects ppp there will
#  be two sessions. One for Login-User and one for Framed-User
#  service type. We only need to take into account the second one.
#
#  The module should be added in the instantiate, authorize and
#  accounting sections.  Make sure that in the authorize
#  section it comes after any module which sets the
#  'check-name' attribute.
#
counter daily {
        filename = ${raddbdir}/db.daily
        key = User-Name
        count-attribute = Acct-Session-Time
        reset = daily
        counter-name = Daily-Session-Time
        check-name = Max-Daily-Session
        allowed-servicetype = Framed-User
        cache-size = 5000
}

# The "always" module is here for debugging purposes. Each
# instance simply returns the same result, always, without
# doing anything.
always fail {
        rcode = fail
}
always reject {
        rcode = reject
}
always ok {
        rcode = ok
        simulcount = 0
        mpp = no
}

#
#  The 'expression' module currently has no configuration.
#
#  This module is useful only for 'xlat'.  To use it,
#  put 'exec' into the 'instantiate' section.  You can then
#  do dynamic translation of attributes like:
#
#  Attribute-Name = `%{expr:2 + 3 + %{exec: uid -u}}`
#
```

```
#  The value of the attribute will be replaced with the output
#  of the program which is executed.  Due to RADIUS protocol
#  limitations, any output over 253 bytes will be ignored.
expr {
}

#
#  The 'digest' module currently has no configuration.
#
#  "Digest" authentication against a Cisco SIP server.
#  See 'doc/rfc/draft-sterman-aaa-sip-00.txt' for details
#  on performing digest authentication for Cisco SIP servers.
#
digest {
}

#
#  Execute external programs
#
#  This module is useful only for 'xlat'.  To use it,
#  put 'exec' into the 'instantiate' section.  You can then
#  do dynamic translation of attributes like:
#
#  Attribute-Name = `%{exec:/path/to/program args}`
#
#  The value of the attribute will be replaced with the output
#  of the program which is executed.  Due to RADIUS protocol
#  limitations, any output over 253 bytes will be ignored.
#
#  The RADIUS attributes from the user request will be placed
#  into environment variables of the executed program, as
#  described in 'doc/variables.txt'
#
exec {
        wait = yes
        input_pairs = request
}

#
#  This is a more general example of the execute module.
#
#  This one is called "echo".
#
#  Attribute-Name = `%{echo:/path/to/program args}`
#
#  If you wish to execute an external program in more than
#  one section (e.g. 'authorize', 'pre_proxy', etc), then it
#  is probably best to define a different instance of the
#  'exec' module for every section.
#
exec echo {
        #
        #  Wait for the program to finish.
        #
        #  If we do NOT wait, then the program is "fire and
        #  forget", and any output attributes from it are ignored.
        #
        #  If we are looking for the program to output
        #  attributes, and want to add those attributes to the
        #  request, then we MUST wait for the program to
        #  finish, and therefore set 'wait=yes'
        #
        # allowed values: {no, yes}
        wait = yes

        #
        #  The name of the program to execute, and it's
        #  arguments.  Dynamic translation is done on this
        #  field, so things like the following example will
        #  work.
        #
        program = "/bin/echo %{User-Name}"

        #
        #  The attributes which are placed into the
        #  environment variables for the program.
        #
        #  Allowed values are:
        #
        #       request        attributes from the request
        #       config         attributes from the configuration items list
        #       reply          attributes from the reply
        #       proxy-request  attributes from the proxy request
        #       proxy-reply    attributes from the proxy reply
        #
        #  Note that some attributes may not exist at some
        #  stages.  e.g. There may be no proxy-reply
        #  attributes if this module is used in the
        #  'authorize' section.
        #
        input_pairs = request

        #
        #  Where to place the output attributes (if any) from
        #  the executed program.  The values allowed, and the
        #  restrictions as to availability, are the same as
        #  for the input_pairs.
        #
        output_pairs = reply

        #
        #  When to execute the program.  If the packet
        #  type does NOT match what's listed here, then
        #  the module does NOT execute the program.
        #
        #  For a list of allowed packet types, see
        #  the 'dictionary' file, and look for VALUEs
        #  of the Packet-Type attribute.
        #
        #  By default, the module executes on ANY packet.
        #  Un-comment out the following line to tell the
        #  module to execute only if an Access-Accept is
        #  being sent to the NAS.
        #
        #packet_type = Access-Accept
}

#  Do server side ip pool management. Should be added in post-auth and
#  accounting sections.
#
#  The module also requires the existance of the Pool-Name
#  attribute. That way the administrator can add the Pool-Name
#  attribute in the user profiles and use different pools
#  for different users. The Pool-Name attribute is a *check* item not
#  a reply item.
#
# Example:
# radiusd.conf: ippool students { [...] }
# users file  : DEFAULT Group == students, Pool-Name := "students"
#
# ********* IF YOU CHANGE THE RANGE PARAMETERS YOU MUST *********
# ********* THEN ERASE THE DB FILES                     *********
#
ippool main_pool {

        #  range-start,range-stop: The start and end ip
        #  addresses for the ip pool
        range-start = 192.168.1.1
        range-stop = 192.168.3.254

        #  netmask: The network mask used for the ip's
        netmask = 255.255.255.0
```

```
		#  cache-size: The gdbm cache size for the db
		#  files. Should be equal to the number of ip's
		#  available in the ip pool
		cache-size = 800

		# session-db: The main db file used to allocate ip's to clients
		session-db = ${raddbdir}/db.ippool

		# ip-index: Helper db index file used in multilink
		ip-index = ${raddbdir}/db.ipindex

		# override: Will this ippool override a Framed-IP-Address already set
		override = no

		# maximum-timeout: If not zero specifies the maximum time in seconds an
		# entry may be active. Default: 0
		maximum-timeout = 0
	}

	# ANSI X9.9 token support.  Not included by default.
	# $INCLUDE  ${confdir}/x99.conf

}

# Instantiation
#
#  This section orders the loading of the modules.  Modules
#  listed here will get loaded BEFORE the later sections like
#  authorize, authenticate, etc. get examined.
#
#  This section is not strictly needed.  When a section like
#  authorize refers to a module, it's automatically loaded and
#  initialized.  However, some modules may not be listed in any
#  of the following sections, so they can be listed here.
#
#  Also, listing modules here ensures that you have control over
#  the order in which they are initalized.  If one module needs
#  something defined by another module, you can list them in order
#  here, and ensure that the configuration will be OK.
#
instantiate {
	#
	#  Allows the execution of external scripts.
	#  The entire command line (and output) must fit into 253 bytes.
	#
	#  e.g. Framed-Pool = `%{exec:/bin/echo foo}`
	exec

	#
	#  The expression module doesn't do authorization,
	#  authentication, or accounting.  It only does dynamic
	#  translation, of the form:
	#
	#	Session-Timeout = `%{expr:2 + 3}`
	#
	#  So the module needs to be instantiated, but CANNOT be
	#  listed in any other section.  See 'doc/rlm_expr' for
	#  more information.
	#
	expr

	#
	# We add the counter module here so that it registers
	# the check-name attribute before any module which sets
	# it
#	daily
}

#  Authorization. First preprocess (hints and huntgroups files),
#  then realms, and finally look in the "users" file.
```

```
#
#  The order of the realm modules will determine the order that
#  we try to find a matching realm.
#
#  Make *sure* that 'preprocess' comes before any realm if you
#  need to setup hints for the remote radius server
authorize {
        #
        #  The preprocess module takes care of sanitizing some bizarre
        #  attributes in the request, and turning them into attributes
        #  which are more standard.
        #
        #  It takes care of processing the 'raddb/hints' and the
        #  'raddb/huntgroups' files.
        #
        #  It also adds the %{Client-IP-Address} attribute to the request.
        preprocess

        #
        #  If you want to have a log of authentication requests,
        #  un-comment the following line, and the 'detail auth_log'
        #  section, above.
#       auth_log

#       attr_filter

        #
        #  The chap module will set 'Auth-Type := CHAP' if we are
        #  handling a CHAP request and Auth-Type has not already been set
        chap

        #
        #  If the users are logging in with an MS-CHAP-Challenge
        #  attribute for authentication, the mschap module will find
        #  the MS-CHAP-Challenge attribute, and add 'Auth-Type := MS-CHAP'
        #  to the request, which will cause the server to then use
        #  the mschap module for authentication.
        mschap

        #
        #  If you have a Cisco SIP server authenticating against
        #  FreeRADIUS, uncomment the following line, and the 'digest'
        #  line in the 'authenticate' section.
        digest

        #
        #  Look for IPASS style 'realm/', and if not found, look for
        #  '@realm', and decide whether or not to proxy, based on
        #  that.
        IPASS

        #
        #  If you are using multiple kinds of realms, you probably
        #  want to set "ignore_null = yes" for all of them.
        #  Otherwise, when the first style of realm doesn't match,
        #  the other styles won't be checked.
        #
        suffix
        ntdomain

        #
        #  This module takes care of EAP-MD5, EAP-TLS, and EAP-LEAP
        #  authentication.
        #
        #  It also sets the EAP-Type attribute in the request
        #  attribute list to the EAP type from the packet.
        eap

        #
        #  Read the 'users' file
        files
        #
        #  Look in an SQL database.  The schema of the database
        #  is meant to mirror the "users" file.
        #
        #  See "Authorization Queries" in sql.conf
#       sql

        #
        #  If you are using /etc/smbpasswd, and are also doing
        #  mschap authentication, the un-comment this line, and
        #  configure the 'etc_smbpasswd' module, above.
#       etc_smbpasswd

        #
        #  The ldap module will set Auth-Type to LDAP if it has not
        #  already been set
        ldap

        #
        #  Enforce daily limits on time spent logged in.
#       daily

        #
        # Use the checkval module
#       checkval
}


#  Authentication.
#
#
#  This section lists which modules are available for authentication.
#  Note that it does NOT mean 'try each module in order'.  It means
#  that a module from the 'authorize' section adds a configuration
#  attribute 'Auth-Type := FOO'.  That authentication type is then
#  used to pick the apropriate module from the list below.
#

#  In general, you SHOULD NOT set the Auth-Type attribute.  The server
#  will figure it out on its own, and will do the right thing.  The
#  most common side effect of erroneously setting the Auth-Type
#  attribute is that one authentication method will work, but the
#  others will not.
#
#  The common reasons to set the Auth-Type attribute by hand
#  is to either forcibly reject the user, or forcibly accept him.
#
authenticate {
        #
        #  PAP authentication, when a back-end database listed
        #  in the 'authorize' section supplies a password.  The
        #  password can be clear-text, or encrypted.
        Auth-Type PAP {
                pap
        }

        #
        #  Most people want CHAP authentication
        #  A back-end database listed in the 'authorize' section
        #  MUST supply a CLEAR TEXT password.  Encrypted passwords
        #  won't work.
        Auth-Type CHAP {
                chap
        }

        #
        #  MSCHAP authentication.
        Auth-Type MS-CHAP {
                mschap
        }
        #
        #  If you have a Cisco SIP server authenticating against
        #  FreeRADIUS, uncomment the following line, and the 'digest'
        #  line in the 'authorize' section.
#       digest

        #
        #  Pluggable Authentication Modules.
#       pam

# kerberos
        Auth-Type Kerberos {
                krb5
        }

        #
        #  See 'man getpwent' for information on how the 'unix'
        #  module checks the users password.  Note that packets
        #  containing CHAP-Password attributes CANNOT be authenticated
        #  against /etc/passwd!  See the FAQ for details.
        #
#       unix

        # Uncomment it if you want to use ldap for authentication
        #
        # Note that this means "check plain-text password against
        # the ldap database", which means that EAP won't work,
        # as it does not supply a plain-text password.
#       Auth-Type LDAP {
#               ldap
#       }
#
        #
        #  Allow EAP authentication.
        eap
}


#
#  Pre-accounting.  Decide which accounting type to use.
#
preacct {
        preprocess

        #
        #  Ensure that we have a semi-unique identifier for every
        #  request, and many NAS boxes are broken.
        acct_unique

        #
        #  Look for IPASS-style 'realm/', and if not found, look for
        #  '@realm', and decide whether or not to proxy, based on
        #  that.
        #
        #  Accounting requests are generally proxied to the same
        #  home server as authentication requests.
#       IPASS
        suffix
#       ntdomain

        #
        #  Read the 'acct_users' file
        files
}

#
#  Accounting.  Log the accounting data.
#
accounting {
        #
        #  Create a 'detail'ed log of the packets.
        #  Note that accounting requests which are proxied
        #  are also logged in the detail file.
        detail
#       daily

        #  Update the wtmp file
        #
        #  If you don't use "radlast", you can delete this line.
        unix

        #
        #  For Simultaneous-Use tracking.
        #
        #  Due to packet losses in the network, the data here
        #  may be incorrect.  There is little we can do about it.
        radutmp
#       sradutmp

        #  Return an address to the IP Pool when we see a stop record.
#       main_pool

        #
        #  Log traffic to an SQL database.
        #
        #  See "Accounting queries" in sql.conf
#       sql


        #  Cisco VoIP specific bulk accounting
#       pgsql-voip

}


#  Session database, used for checking Simultaneous-Use. Either the radutmp
#  or rlm_sql module can handle this.
#  The rlm_sql module is *much* faster
session {
        radutmp

        #
        #  See "Simultaneous Use Checking Querie" in sql.conf
#       sql
}


#  Post-Authentication
#  Once we KNOW that the user has been authenticated, there are
#  additional steps we can take.
post-auth {
        #  Get an address from the IP Pool.
#       main_pool

        #
        #  If you want to have a log of authentication replies,
        #  un-comment the following line, and the 'detail reply_log'
        #  section, above.
#       reply_log

        #
        #  After authenticating the user, do another SQL qeury.
        #
        #  See "Authentication Logging Queries" in sql.conf
#       sql

        #
        #  Access-Reject packets are sent through the REJECT sub-section
        #  of the post-auth section.
        #
#       Post-Auth-Type REJECT {
```

```
#               insert-module-name-here
#       }

}

#
#  When the server decides to proxy a request to a home server,
#  the proxied request is first passed through the pre-proxy
#  stage.  This stage can re-write the request, or decide to
#  cancel the proxy.
#
#  Only a few modules currently have this method.
#
pre-proxy {
#       attr_rewrite

        #  If you want to have a log of packets proxied to a home
        #  server, un-comment the following line, and the
        #  'detail pre_proxy_log' section, above.
#       pre_proxy_log
}

#
#  When the server receives a reply to a request it proxied
#  to a home server, the request may be massaged here, in the
#  post-proxy stage.
#
post-proxy {
        #

        #  If you want to have a log of replies from a home server,
        #  un-comment the following line, and the 'detail post_proxy_log'
        #  section, above.
#       post_proxy_log

#       attr_rewrite

        #  Uncomment the following line if you want to filter replies from
        #  remote proxies based on the rules defined in the 'attrs' file.

#       attr_filter

        #
        #  If you are proxying LEAP, you MUST configure the EAP
        #  module, and you MUST list it here, in the post-proxy
        #  stage.
        #
        #  You MUST also use the 'nostrip' option in the 'realm'
        #  configuration.  Otherwise, the User-Name attribute
        #  in the proxied request will not match the user name
        #  hidden inside of the EAP packet, and the end server will
        #  reject the EAP request.
        #
        eap
}
```

users:

```
DEFAULT EAP-Type == EAP-TLS, Auth-Type := Reject

DEFAULT         Auth-Type = Kerberos
```

La configurazione di questo file nega l’accesso all’autenticazione EAP-TLS (certificate digitali) e consente solamente l'autenticazione Kerberos

Configurazione di freeradius autenticazione utenti locale

Sono riporttidi seguito le configurazioni di *radiusd.conf* e *users* nello scenario in cui gli utenti siano direttamente inseriti sul radius server nel database delle password unix.

radiusd.conf:

```
##
## radiusd.conf       -- FreeRADIUS server configuration file.
##
##      http://www.freeradius.org/
##      $Id: radiusd.conf.in,v 1.188 2004/05/13 20:10:19 pnixon Exp $
##

#       The location of other config files and
#       logfiles are declared in this file
#
#       Also general configuration for modules can be done
#       in this file, it is exported through the API to
#       modules that ask for it.
#
#       The configuration variables defined here are of the form ${foo}
#       They are local to this file, and do not change from request to
#       request.
#
#       The per-request variables are of the form %{Attribute-Name}, and
#       are taken from the values of the attribute in the incoming
#       request.  See 'doc/variables.txt' for more information.

prefix = /usr
exec_prefix = /usr
sysconfdir = /etc
localstatedir = /var
sbindir = /usr/sbin
logdir = ${localstatedir}/log/radius
raddbdir = ${sysconfdir}/raddb
radacctdir = ${logdir}/radacct

#  Location of config and logfiles.
confdir = ${raddbdir}
run_dir = ${localstatedir}/run/radiusd

#
#  The logging messages for the server are appended to the
#  tail of this file.
#
log_file = ${logdir}/radius.log

#
# libdir: Where to find the rlm_* modules.
#
#   This should be automatically set at configuration time.
#
#   If the server builds and installs, but fails at execution time
#   with an 'undefined symbol' error, then you can use the libdir
#   directive to work around the problem.
#
```

```
#   The cause is usually that a library has been installed on your
#   system in a place where the dynamic linker CANNOT find it.  When
#   executing as root (or another user), your personal environment MAY
#   be set up to allow the dynamic linker to find the library.  When
#   executing as a daemon, FreeRADIUS MAY NOT have the same
#   personalized configuration.
#
#   To work around the problem, find out which library contains that symbol,
#   and add the directory containing that library to the end of 'libdir',
#   with a colon separating the directory names.  NO spaces are allowed.
#
#   e.g. libdir = /usr/local/lib:/opt/package/lib
#
#   You can also try setting the LD_LIBRARY_PATH environment variable
#   in a script which starts the server.
#
#   If that does not work, then you can re-configure and re-build the
#   server to NOT use shared libraries, via:
#
#       ./configure --disable-shared
#       make
#       make install
#
libdir = /usr/lib

#  pidfile: Where to place the PID of the RADIUS server.
#
#  The server may be signalled while it's running by using this
#  file.
#
#  This file is written when ONLY running in daemon mode.
#
#  e.g.:  kill -HUP `cat /var/run/radiusd/radiusd.pid`
#
pidfile = ${run_dir}/radiusd.pid


# user/group: The name (or #number) of the user/group to run radiusd as.
#
#   If these are commented out, the server will run as the user/group
#   that started it.  In order to change to a different user/group, you
#   MUST be root ( or have root privleges ) to start the server.
#
#   We STRONGLY recommend that you run the server with as few permissions
#   as possible.  That is, if you're not using shadow passwords, the
#   user and group items below should be set to 'nobody'.
#
#    On SCO (ODT 3) use "user = nouser" and "group = nogroup".
#
#  NOTE that some kernels refuse to setgid(group) when the value of
#  (unsigned)group is above 60000; don't use group nobody on these systems!
#
#  On systems with shadow passwords, you might have to set 'group = shadow'
#  for the server to be able to read the shadow password file.  If you can
#  authenticate users while in debug mode, but not in daemon mode, it may be
#  that the debugging mode server is running as a user that can read the
#  shadow info, and the user listed below can not.
#
user = root
group = radiusd

#  max_request_time: The maximum time (in seconds) to handle a request.
#
#  Requests which take more time than this to process may be killed, and
#  a REJECT message is returned.
#
#  WARNING: If you notice that requests take a long time to be handled,
#  then this MAY INDICATE a bug in the server, in one of the modules
#  used to handle a request, OR in your local configuration.
#
#  This problem is most often seen when using an SQL database.  If it takes
#  more than a second or two to receive an answer from the SQL database,
#  then it probably means that you haven't indexed the database.  See your
#  SQL server documentation for more information.
#
#  Useful range of values: 5 to 120
#
max_request_time = 30

#  delete_blocked_requests: If the request takes MORE THAN 'max_request_time'
#  to be handled, then maybe the server should delete it.
#
#  If you're running in threaded, or thread pool mode, this setting
#  should probably be 'no'.  Setting it to 'yes' when using a threaded
#  server MAY cause the server to crash!
#
delete_blocked_requests = no

#  cleanup_delay: The time to wait (in seconds) before cleaning up
#  a reply which was sent to the NAS.
#
#  The RADIUS request is normally cached internally for a short period
#  of time, after the reply is sent to the NAS.  The reply packet may be
#  lost in the network, and the NAS will not see it.  The NAS will then
#  re-send the request, and the server will respond quickly with the
#  cached reply.
#
#  If this value is set too low, then duplicate requests from the NAS
#  MAY NOT be detected, and will instead be handled as seperate requests.
#
#  If this value is set too high, then the server will cache too many
#  requests, and some new requests may get blocked.  (See 'max_requests'.)
#
#  Useful range of values: 2 to 10
#
cleanup_delay = 5

#  max_requests: The maximum number of requests which the server keeps
#  track of.  This should be 256 multiplied by the number of clients.
#  e.g. With 4 clients, this number should be 1024.
#
#  If this number is too low, then when the server becomes busy,
#  it will not respond to any new requests, until the 'cleanup_delay'
#  time has passed, and it has removed the old requests.
#
#  If this number is set too high, then the server will use a bit more
#  memory for no real benefit.
#
#  If you aren't sure what it should be set to, it's better to set it
#  too high than too low.  Setting it to 1000 per client is probably
#  the highest it should be.
#
#  Useful range of values: 256 to infinity
#
max_requests = 1024

#  bind_address:  Make the server listen on a particular IP address, and
#  send replies out from that address.  This directive is most useful
#  for machines with multiple IP addresses on one interface.
#
#  It can either contain "*", or an IP address, or a fully qualified
#  Internet domain name.  The default is "*"
#
#  As of 1.0, you can also use the "listen" directive.  See below for
#  more information.
#
bind_address = *

#  port: Allows you to bind FreeRADIUS to a specific port.
#
#  The default port that most NAS boxes use is 1645, which is historical.
#  RFC 2138 defines 1812 to be the new port.  Many new servers and
#  NAS boxes use 1812, which can create interoperability problems.
#
#  The port is defined here to be 0 so that the server will pick up
#  the machine's local configuration for the radius port, as defined
#  in /etc/services.
#
#  If you want to use the default RADIUS port as defined on your server,
#  (usually through 'grep radius /etc/services') set this to 0 (zero).
#
#  A port given on the command-line via '-p' over-rides this one.
#
#  As of 1.0, you can also use the "listen" directive.  See below for
#  more information.
#
port = 0

#
#  By default, the server uses "bind_address" to listen to all IP's
#  on a machine, or just one IP.  The "port" configuration is used
#  to select the authentication port used when listening on those
#  addresses.
#
#  If you want the server to listen on additional addresses, you can
#  use the "listen" section.  A sample section (commented out) is included
#  below.  This "listen" section duplicates the functionality of the
#  "bind_address" and "port" configuration entries, but it only listens
#  for authentication packets.
#
#  If you comment out the "bind_address" and "port" configuration entries,
#  then it becomes possible to make the server accept only accounting,
#  or authentication packets.  Previously, it always listened for both
#  types of packets, and it was impossible to make it listen for only
#  one type of packet.
#
#listen {
        #  IP address on which to listen.
        #  Allowed values are:
        #       dotted quad (1.2.3.4)
        #       hostname    (radius.example.com)
        #       wildcard    (*)
#       ipaddr = *

        #  Port on which to listen.
        #  Allowed values are:
        #       integer port number (1812)
        #       0 means "use /etc/services for the proper port"
#       port = 0

        #  Type of packets to listen for.
        #  Allowed values are:
        #       auth    listen for authentication packets
        #       acct    listen for accounting packets
        #
#       type = auth
#}


#  hostname_lookups: Log the names of clients or just their IP addresses
#  e.g., www.freeradius.org (on) or 206.47.27.232 (off).
#
#  The default is 'off' because it would be overall better for the net
#  if people had to knowingly turn this feature on, since enabling it
#  means that each client request will result in AT LEAST one lookup
#  request to the nameserver.   Enabling hostname_lookups will also
#  mean that your server may stop randomly for 30 seconds from time
#  to time, if the DNS requests take too long.
#
#  Turning hostname lookups off also means that the server won't block
#  for 30 seconds, if it sees an IP address which has no name associated
#  with it.
#
```

```
#  allowed values: {no, yes}
#
hostname_lookups = no

#  Core dumps are a bad thing.  This should only be set to 'yes'
#  if you're debugging a problem with the server.
#
#  allowed values: {no, yes}
#
allow_core_dumps = no

#  Regular expressions
#
#  These items are set at configure time.  If they're set to "yes",
#  then setting them to "no" turns off regular expression support.
#
#  If they're set to "no" at configure time, then setting them to "yes"
#  WILL NOT WORK.  It will give you an error.
#
regular_expressions    = yes
extended_expressions   = yes

#  Log the full User-Name attribute, as it was found in the request.
#
# allowed values: {no, yes}
#
log_stripped_names = no

#  Log authentication requests to the log file.
#
#  allowed values: {no, yes}
#
log_auth = yes

#  Log passwords with the authentication requests.
#  log_auth_badpass  - logs password if it's rejected
#  log_auth_goodpass - logs password if it's correct
#
#  allowed values: {no, yes}
#
log_auth_badpass = no
log_auth_goodpass = no

# usercollide:  Turn "username collision" code on and off.  See the
# "doc/duplicate-users" file
#
#  WARNING
#  !!!!!!!  Setting this to "yes" may result in the server behaving
#  !!!!!!!  strangely.  The "username collision" code will ONLY work
#  !!!!!!!  with clear-text passwords.  Even then, it may not do what
#  !!!!!!!  you want, or what you expect.
#  !!!!!!!
#  !!!!!!!  We STRONGLY RECOMMEND that you do not use this feature,
#  !!!!!!!  and that you find another way of acheiving the same goal.
#  !!!!!!!
#  !!!!!!!  e,g. module fail-over.  See 'doc/configurable_failover'
#  WARNING
#
usercollide = no

# lower_user / lower_pass:
# Lower case the username/password "before" or "after"
# attempting to authenticate.
#
#  If "before", the server will first modify the request and then try
#  to auth the user.  If "after", the server will first auth using the
#  values provided by the user.  If that fails it will reprocess the
#  request after modifying it as you specify below.
#
#  This is as close as we can get to case insensitivity.  It is the
#  admin's job to ensure that the username on the auth db side is
#  *also* lowercase to make this work
#
# Default is 'no' (don't lowercase values)
# Valid values = "before" / "after" / "no"
#
lower_user = no
lower_pass = no

# nospace_user / nospace_pass:
#
#  Some users like to enter spaces in their username or password
#  incorrectly.  To save yourself the tech support call, you can
#  eliminate those spaces here:
#
# Default is 'no' (don't remove spaces)
# Valid values = "before" / "after" / "no" (explanation above)
#
nospace_user = no
nospace_pass = no

#  The program to execute to do concurrency checks.
checkrad = ${sbindir}/checkrad

# SECURITY CONFIGURATION
#
#  There may be multiple methods of attacking on the server.  This
#  section holds the configuration items which minimize the impact
#  of those attacks
#
security {
        #
        #  max_attributes: The maximum number of attributes
        #  permitted in a RADIUS packet.  Packets which have MORE
        #  than this number of attributes in them will be dropped.
        #
        #  If this number is set too low, then no RADIUS packets
        #  will be accepted.
        #
        #  If this number is set too high, then an attacker may be
        #  able to send a small number of packets which will cause
        #  the server to use all available memory on the machine.
        #
        #  Setting this number to 0 means "allow any number of attributes"
        max_attributes = 200

        #
        #  delayed_reject: When sending an Access-Reject, it can be
        #  delayed for a few seconds.  This may help slow down a DoS
        #  attack.  It also helps to slow down people trying to brute-force
        #  crack a users password.
        #
        #  Setting this number to 0 means "send rejects immediately"
        #
        #  If this number is set higher than 'cleanup_delay', then the
        #  rejects will be sent at 'cleanup_delay' time, when the request
        #  is deleted from the internal cache of requests.
        #
        #  Useful ranges: 1 to 5
        reject_delay = 1

        #
        #  status_server: Whether or not the server will respond
        #  to Status-Server requests.
        #
        #  Normally this should be set to "no", because they're useless.
        #  See: http://www.freeradius.org/rfc/rfc2865.html#Keep-Alives
        #
        #  However, certain NAS boxes may require them.
        #
        #  When sent a Status-Server message, the server responds with
        #  an Access-Accept packet, containing a Reply-Message attribute,
        #  which is a string describing how long the server has been
        #  running.
        #
        status_server = no
}

# PROXY CONFIGURATION
#
#  proxy_requests: Turns proxying of RADIUS requests on or off.
#
#  The server has proxying turned on by default.  If your system is NOT
#  set up to proxy requests to another server, then you can turn proxying
#  off here.  This will save a small amount of resources on the server.
#
#  If you have proxying turned off, and your configuration files say
#  to proxy a request, then an error message will be logged.
#
#  To disable proxying, change the "yes" to "no", and comment the
#  $INCLUDE line.
#
#  allowed values: {no, yes}
#
proxy_requests  = yes
$INCLUDE  ${confdir}/proxy.conf


# CLIENTS CONFIGURATION
#
#  Client configuration is defined in "clients.conf".
#

#  The 'clients.conf' file contains all of the information from the old
#  'clients' and 'naslist' configuration files.  We recommend that you
#  do NOT use 'client's or 'naslist', although they are still
#  supported.
#
#  Anything listed in 'clients.conf' will take precedence over the
#  information from the old-style configuration files.
#
$INCLUDE  ${confdir}/clients.conf


# SNMP CONFIGURATION
#
#  Snmp configuration is only valid if SNMP support was enabled
#  at compile time.
#
#  To enable SNMP querying of the server, set the value of the
#  'snmp' attribute to 'yes'
#
snmp    = no
$INCLUDE  ${confdir}/snmp.conf


# THREAD POOL CONFIGURATION
#
#  The thread pool is a long-lived group of threads which
#  take turns (round-robin) handling any incoming requests.
#
#  You probably want to have a few spare threads around,
#  so that high-load situations can be handled immediately.  If you
#  don't have any spare threads, then the request handling will
#  be delayed while a new thread is created, and added to the pool.
#
#  You probably don't want too many spare threads around,
#  otherwise they'll be sitting there taking up resources, and
#  not doing anything productive.
#
#  The numbers given below should be adequate for most situations.
#
thread pool {
        #  Number of servers to start initially --- should be a reasonable
        #  ballpark figure.
        start_servers = 5

        #  Limit on the total number of servers running.
        #
        #  If this limit is ever reached, clients will be LOCKED OUT, so it
        #  should NOT BE SET TOO LOW.  It is intended mainly as a brake to
        #  keep a runaway server from taking the system with it as it spirals
        #  down...
        #
        #  You may find that the server is regularly reaching the
        #  'max_servers' number of threads, and that increasing
        #  'max_servers' doesn't seem to make much difference.
        #
        #  If this is the case, then the problem is MOST LIKELY that
        #  your back-end databases are taking too long to respond, and
        #  are preventing the server from responding in a timely manner.
        #
        #  The solution is NOT do keep increasing the 'max_servers'
        #  value, but instead to fix the underlying cause of the
        #  problem: slow database, or 'hostname_lookups=yes'.
        #
        #  For more information, see 'max_request_time', above.
        #
        max_servers = 32

        #  Server-pool size regulation.  Rather than making you guess
        #  how many servers you need, FreeRADIUS dynamically adapts to
        #  the load it sees, that is, it tries to maintain enough
        #  servers to handle the current load, plus a few spare
        #  servers to handle transient load spikes.
        #
        #  It does this by periodically checking how many servers are
        #  waiting for a request.  If there are fewer than
        #  min_spare_servers, it creates a new spare.  If there are
        #  more than max_spare_servers, some of the spares die off.
        #  The default values are probably OK for most sites.
        #
        min_spare_servers = 3
        max_spare_servers = 10

        #  There may be memory leaks or resource allocation problems with
        #  the server.  If so, set this value to 300 or so, so that the
        #  resources will be cleaned up periodically.
        #
        #  This should only be necessary if there are serious bugs in the
        #  server which have not yet been fixed.
        #
        #  '0' is a special value meaning 'infinity', or 'the servers never
        #  exit'
        max_requests_per_server = 0
}

# MODULE CONFIGURATION
#
#  The names and configuration of each module is located in this section.
#
#  After the modules are defined here, they may be referred to by name,
#  in other sections of this configuration file.
#
modules {
        #
        #  Each module has a configuration as follows:
        #
        #       name [ instance ] {
        #               config_item = value
        #               ...
        #       }
        #
        #  The 'name' is used to load the 'rlm_name' library
        #  which implements the functionality of the module.
        #
        #  The 'instance' is optional.  To have two different instances
        #  of a module, it first must be referred to by 'name'.
        #  The different copies of the module are then created by
        #  inventing two 'instance' names, e.g. 'instance1' and 'instance2'
        #
        #  The instance names can then be used in later configuration
        #  INSTEAD of the original 'name'.  See the 'radutmp' configuration
        #  below for an example.
        #

        # PAP module to authenticate users based on their stored password
        #
        #  Supports multiple encryption schemes
        #  clear: Clear text
        #  crypt: Unix crypt
        #    md5: MD5 ecnryption
        #   sha1: SHA1 encryption.
        #  DEFAULT: crypt
        pap {
                encryption_scheme = crypt
        }

        # CHAP module
        #
        #  To authenticate requests containing a CHAP-Password attribute.
        #
        chap {
                authtype = CHAP
        }

        # Pluggable Authentication Modules
        #
        #  For Linux, see:
        #       http://www.kernel.org/pub/linux/libs/pam/index.html
        #
        #  WARNING: On many systems, the system PAM libraries have
        #           memory leaks!  We STRONGLY SUGGEST that you do not
        #           use PAM for authentication, due to those memory leaks.
        #
        pam {
                #
                #  The name to use for PAM authentication.
                #  PAM looks in /etc/pam.d/${pam_auth_name}
                #  for it's configuration.  See 'redhat/radiusd-pam'
                #  for a sample PAM configuration file.
                #
                #  Note that any Pam-Auth attribute set in the 'authorize'
                #  section will over-ride this one.
                #
                pam_auth = radiusd
        }
        # Kerberos
        krb5 {
                keytab = /etc/krb5.keytab
                #service_principal =
        }

        # Unix /etc/passwd style authentication
        #
        unix {
                #
                #  Cache /etc/passwd, /etc/shadow, and /etc/group
                #
                #  The default is to NOT cache them.
                #
                #  For FreeBSD and NetBSD, you do NOT want to enable
                #  the cache, as it's password lookups are done via a
                #  database, so set this value to 'no'.
                #
```

```
		#  Some systems (e.g. RedHat Linux with pam_pwbd) can
		#  take *seconds* to check a password, when th passwd
		#  file containing 1000's of entries.  For those systems,
		#  you should set the cache value to 'yes', and set
		#  the locations of the 'passwd', 'shadow', and 'group'
		#  files, below.
		#
		# allowed values: {no, yes}
		cache = no

		# Reload the cache every 600 seconds (10mins). 0 to disable.
		cache_reload = 600

		#
		#  Define the locations of the normal passwd, shadow, and
		#  group files.
		#
		#  'shadow' is commented out by default, because not all
		#  systems have shadow passwords.
		#
		#  To force the module to use the system password functions,
		#  instead of reading the files, leave the following entries
		#  commented out.
		#
		#  This is required for some systems, like FreeBSD,
		#  and Mac OSX.
		#
		#	passwd = /etc/passwd
		shadow = /etc/shadow
		#	group = /etc/group

		#
		#  The location of the "wtmp" file.
		#  This should be moved to it's own module soon.
		#
		#  The only use for 'radlast'.  If you don't use
		#  'radlast', then you can comment out this item.
		#
		radwtmp = ${logdir}/radwtmp
	}

	#  Extensible Authentication Protocol
	#
	#  For all EAP related authentications.
	#  Now in another file, because it is very large.
	#
$INCLUDE ${confdir}/eap.conf

	# Microsoft CHAP authentication
	#
	#  This module supports MS-CHAP and MS-CHAPv2 authentication.
	#  It also enforces the SMB-Account-Ctrl attribute.
	#
	mschap {
		#
		#  As of 0.9, the mschap module does NOT support
		#  reading from /etc/smbpasswd.
		#
		#  If you are using /etc/smbpasswd, see the 'passwd'
		#  module for an example of how to use /etc/smbpasswd

		# authtype value, if present, will be used
		# to overwrite (or add) Auth-Type during
		# authorization. Normally should be MS-CHAP
		authtype = MS-CHAP

		# if use_mppe is not set to no mschap will
		# add MS-CHAP-MPPE-Keys for MS-CHAPv1 and
		# MS-MPPE-Recv-Key/MS-MPPE-Send-Key for MS-CHAPv2
		#
		#use_mppe = no
		# if mppe is enabled require_encryption makes
		# encryption moderate
		#
		#require_encryption = yes

		# require_strong always requires 128 bit key
		# encryption
		#
		#require_strong = yes

		# Windows sends us a username in the form of
		# DOMAIN\user, but sends the challenge response
		# based on only the user portion.  This hack
		# corrects for that incorrect behavior.
		#
		#with_ntdomain_hack = no

		# The module can perform authentication itself, OR
		# use a Windows Domain Controller.  This configuration
		# directive tells the module to call the ntlm_auth
		# program, which will do the authentication, and return
		# the NT-Key.  Note that you MUST have "winbindd" and
		# "nmbd" running on the local machine for ntlm_auth
		# to work.  See the ntlm_auth program documentation
		# for details.
		#
		# Be VERY careful when editing the following line!
		#
		#ntlm_auth = "/path/to/ntlm_auth --request-nt-key --username=%{Stripped-
User-Name:-%{User-Name:-None}} --challenge=%{mschap:Challenge:-00} --nt-
response=%{mschap:NT-Response:-00}"
	}

	# Lightweight Directory Access Protocol (LDAP)
	#
	#  This module definition allows you to use LDAP for
	#  authorization and authentication (Auth-Type := LDAP)
	#
	#  See doc/rlm_ldap for description of configuration options
	#  and sample authorize{} and authenticate{} blocks
	ldap {
		#server = "server.domain.infn.it"
		# identity = "cn=admin,o=My Org,c=UA"
		# password = mypass
		#basedn = "ou=people,o=something,o=infn,c=it"
		#filter = "(uid=%{Stripped-User-Name:-%{User-Name}})"
		# base_filter = "(objectclass=radiusprofile)"

		# set this to 'yes' to use TLS encrypted connections
		# to the LDAP database by using the StartTLS extended
		# operation.
		# The StartTLS operation is supposed to be used with normal
		# ldap connections instead of using ldaps (port 689) connections
		start_tls = no

		# tls_cacertfile	= /path/to/cacert.pem
		# tls_cacertdir		= /path/to/ca/dir/
		# tls_certfile		= /path/to/radius.crt
		# tls_keyfile		= /path/to/radius.key
		# tls_randfile		= /path/to/rnd
		# tls_require_cert	= "demand"

		# default_profile = "cn=radprofile,ou=dialup,o=My Org,c=UA"
		# profile_attribute = "radiusProfileDn"
		#access_attr = "dialupAccess"

		# Mapping of RADIUS dictionary attributes to LDAP
		# directory attributes.
		dictionary_mapping = ${raddbdir}/ldap.attrmap
		ldap_connections_number = 5

		#
		# NOTICE: The password_header directive is NOT case insensitive
		#
		# password_header = "{clear}"
		#
		#  The server can usually figure this out on its own, and pull
		#  the correct User-Password or NT-Password from the database.
		#
		#  Note that NT-Passwords MUST be stored as a 32-digit hex
		#  string, and MUST start off with "0x", such as:
		#
		#	0x000102030405060708090a0b0c0d0e0f
		#
		#  Without the leading "0x", NT-Passwords will not work.
		#  This goes for NT-Passwords stored in SQL, too.
		#
		# password_attribute = userPassword
		# groupname_attribute = cn
		# groupmembership_filter = "(|(&(objectClass=GroupOfNames)(member=%{Ldap-
UserDn}))(&(objectClass=GroupOfUniqueNames)(uniquemember=%{Ldap-UserDn})))"
		# groupmembership_attribute = radiusGroupName
		timeout = 4
		timelimit = 3
		net_timeout = 1
		# compare_check_items = yes
		# do_xlat = yes
		# access_attr_used_for_allow = yes
	}

	# passwd module allows to do authorization via any passwd-like
	# file and to extract any attributes from these modules
	#
	# parameters are:
	#   filename - path to filename
	#   format - format for filename record. This parameters
	#            correlates record in the passwd file and RADIUS
	#            attributes.
	#
	#            Field marked as '*' is key field. That is, the parameter
	#            with this name from the request is used to search for
	#            the record from passwd file
	#            Attribute marked as '=' is added to reply_itmes instead
	#            of default configure_itmes
	#           Attribute marked as '~' is added to request_items
	#
	#            Field marked as ',' may contain a comma separated list
	#            of attributes.
	#   authtype - if record found this Auth-Type is used to authenticate
	#            user
	#   hashsize - hashtable size. If 0 or not specified records are not
	#            stored in memory and file is red on every request.
	#   allowmultiplekeys - if few records for every key are allowed
	#   ignorenislike - ignore NIS-related records
	#   delimiter - symbol to use as a field separator in passwd file,
	#            for format ':' symbol is always used. '\0', '\n' are
	 #           not allowed
	#

	#  An example configuration for using /etc/smbpasswd.
	#
	#passwd etc_smbpasswd {
	#	filename = /etc/smbpasswd
	#	format = "*User-Name::LM-Password:NT-Password:SMB-Account-CTRL-TEXT::"
	#	authtype = MS-CHAP
	#	hashsize = 100
	#	ignorenislike = no
	#	allowmultiplekeys = no
	#}
```

```
#  Similar configuration, for the /etc/group file. Adds a Group-Name
#  attribute for every group that the user is member of.
#
#passwd etc_group {
#       filename = /etc/group
#       format = "=Group-Name:::*,User-Name"
#       hashsize = 50
#       ignorenislike = yes
#       allowmultiplekeys = yes
#       delimiter = ":"
#}

# Realm module, for proxying.
#
#  You can have multiple instances of the realm module to
#  support multiple realm syntaxs at the same time.  The
#  search order is defined by the order in the authorize and
#  preacct sections.
#
#  Four config options:
#       format         -  must be 'prefix' or 'suffix'
#       delimiter      -  must be a single character
#       ignore_default -  set to 'yes' or 'no'
#        ignore_null    -  set to 'yes' or 'no'
#
#  ignore_default and ignore_null can be set to 'yes' to prevent
#  the module from matching against DEFAULT or NULL realms.  This
#  may be useful if you have have multiple instances of the
#  realm module.
#
#  They both default to 'no'.
#

#  'realm/username'
#
#  Using this entry, IPASS users have their realm set to "IPASS".
realm IPASS {
        format = prefix
        delimiter = "/"
        ignore_default = no
        ignore_null = no
}

#  'username@realm'
#
realm suffix {
        format = suffix
        delimiter = "@"
        ignore_default = no
        ignore_null = no
}

#  'username%realm'
#
realm realmpercent {
        format = suffix
        delimiter = "%"
        ignore_default = no
        ignore_null = no
}

#
#  'domain\user'
#
realm ntdomain {
        format = prefix
        delimiter = "\\"
        ignore_default = no
        ignore_null = no
}
```

```
        #  A simple value checking module
        #
        #  It can be used to check if an attribute value in the request
        #  matches a (possibly multi valued) attribute in the check
        #  items This can be used for example for caller-id
        #  authentication.  For the module to run, both the request
        #  attribute and the check items attribute must exist
        #
        #  i.e.
        #  A user has an ldap entry with 2 radiusCallingStationId
        #  attributes with values "12345678" and "12345679".  If we
        #  enable rlm_checkval, then any request which contains a
        #  Calling-Station-Id with one of those two values will be
        #  accepted.  Requests with other values for
        #  Calling-Station-Id will be rejected.
        #
        #  Regular expressions in the check attribute value are allowed
        #  as long as the operator is '=~'
        #
        checkval {
                # The attribute to look for in the request
                item-name = Calling-Station-Id

                # The attribute to look for in check items. Can be multi valued
                check-name = Calling-Station-Id

                # The data type. Can be
                # string,integer,ipaddr,date,abinary,octets
                data-type = string

                # If set to yes and we dont find the item-name attribute in the
                # request then we send back a reject
                # DEFAULT is no
                #notfound-reject = no
        }

        #  rewrite arbitrary packets.  Useful in accounting and authorization.
        #
        #
        #  The module can also use the Rewrite-Rule attribute. If it
        #  is set and matches the name of the module instance, then
        #  that module instance will be the only one which runs.
        #
        #  Also if new_attribute is set to yes then a new attribute
        #  will be created containing the value replacewith and it
        #  will be added to searchin (packet, reply, proxy, proxy_reply or config).
        # searchfor,ignore_case and max_matches will be ignored in that case.
        #
        # Backreferences are supported: %{0} will contain the string the whole match
        # and %{1} to %{8} will contain the contents of the 1st to the 8th parentheses
        #
        # If max_matches is greater than one the backreferences will correspond to the
        # first match

        #
        #attr_rewrite sanecallerid {
        #       attribute = Called-Station-Id
                # may be "packet", "reply", "proxy", "proxy_reply" or "config"
        #       searchin = packet
        #       searchfor = "[+ ]"
        #       replacewith = ""
        #       ignore_case = no
        #       new_attribute = no
        #       max_matches = 10
        #       ## If set to yes then the replace string will be appended to the original
string
        #       append = no
        #}

        # Preprocess the incoming RADIUS request, before handing it off
        # to other modules.
```

```
#
#  This module processes the 'huntgroups' and 'hints' files.
#  In addition, it re-writes some weird attributes created
#  by some NASes, and converts the attributes into a form which
#  is a little more standard.
#
preprocess {
        huntgroups = ${confdir}/huntgroups
        hints = ${confdir}/hints

        # This hack changes Ascend's wierd port numberings
        # to standard 0-??? port numbers so that the "+" works
        # for IP address assignments.
        with_ascend_hack = no
        ascend_channels_per_line = 23

        # Windows NT machines often authenticate themselves as
        # NT_DOMAIN\username
        #
        # If this is set to 'yes', then the NT_DOMAIN portion
        # of the user-name is silently discarded.
        #
        # This configuration entry SHOULD NOT be used.
        # See the "realms" module for a better way to handle
        # NT domains.
        with_ntdomain_hack = no

        # Specialix Jetstream 8500 24 port access server.
        #
        # If the user name is 10 characters or longer, a "/"
        # and the excess characters after the 10th are
        # appended to the user name.
        #
        # If you're not running that NAS, you don't need
        # this hack.
        with_specialix_jetstream_hack = no

        # Cisco sends it's VSA attributes with the attribute
        # name *again* in the string, like:
        #
        #   H323-Attribute = "h323-attribute=value".
        #
        # If this configuration item is set to 'yes', then
        # the redundant data in the the attribute text is stripped
        # out.  The result is:
        #
        #  H323-Attribute = "value"
        #
        # If you're not running a Cisco NAS, you don't need
        # this hack.
        with_cisco_vsa_hack = no
}

# Livingston-style 'users' file
#
files {
        usersfile = ${confdir}/users
        acctusersfile = ${confdir}/acct_users

        #  If you want to use the old Cistron 'users' file
        #  with FreeRADIUS, you should change the next line
        #  to 'compat = cistron'.  You can the copy your 'users'
        #  file from Cistron.
        compat = no
}

# Write a detailed log of all accounting records received.
#
detail {
        #  Note that we do NOT use NAS-IP-Address here, as
        #  that attribute MAY BE from the originating NAS, and
        #  NOT from the proxy which actually sent us the
        #  request.  The Client-IP-Address attribute is ALWAYS
        #  the address of the client which sent us the
        #  request.
        #
        #  The following line creates a new detail file for
        #  every radius client (by IP address or hostname).
        #  In addition, a new detail file is created every
        #  day, so that the detail file doesn't have to go
        #  through a 'log rotation'
        #
        #  If your detail files are large, you may also want
        #  to add a ':%H' (see doc/variables.txt) to the end
        #  of it, to create a new detail file every hour, e.g.:
        #
        #   ..../detail-%Y%m%d:%H
        #
        #  This will create a new detail file for every hour.
        #
        detailfile = ${radacctdir}/%{Client-IP-Address}/detail-%Y%m%d

        #
        #  The Unix-style permissions on the 'detail' file.
        #
        #  The detail file often contains secret or private
        #  information about users.  So by keeping the file
        #  permissions restrictive, we can prevent unwanted
        #  people from seeing that information.
        detailperm = 0600
}

#
#  Many people want to log authentication requests.
#  Rather than modifying the server core to print out more
#  messages, we can use a different instance of the 'detail'
#  module, to log the authentication requests to a file.
#
#  You will also need to un-comment the 'auth_log' line
#  in the 'authorize' section, below.
#
# detail auth_log {
        # detailfile = ${radacctdir}/%{Client-IP-Address}/auth-detail-%Y%m%d

        #
        #  This MUST be 0600, otherwise anyone can read
        #  the users passwords!
        # detailperm = 0600
# }

#
#  This module logs authentication reply packets sent
#  to a NAS.  Both Access-Accept and Access-Reject packets
#  are logged.
#
#  You will also need to un-comment the 'reply_log' line
#  in the 'post-auth' section, below.
#
# detail reply_log {
        # detailfile = ${radacctdir}/%{Client-IP-Address}/reply-detail-%Y%m%d

        #
        #  This MUST be 0600, otherwise anyone can read
        #  the users passwords!
        # detailperm = 0600
# }

#
#  This module logs packets proxied to a home server.
#
#  You will also need to un-comment the 'pre_proxy_log' line
#  in the 'pre-proxy' section, below.
```

```
        #
        # detail pre_proxy_log {
               # detailfile = ${radacctdir}/%{Client-IP-Address}/pre-proxy-detail-%Y%m%d

               #
               #  This MUST be 0600, otherwise anyone can read
               #  the users passwords!
               # detailperm = 0600
        # }

        #
        #  This module logs response packets from a home server.
        #
        #  You will also need to un-comment the 'post_proxy_log' line
        #  in the 'post-proxy' section, below.
        #
        # detail post_proxy_log {
               # detailfile = ${radacctdir}/%{Client-IP-Address}/post-proxy-detail-%Y%m%d

               #
               #  This MUST be 0600, otherwise anyone can read
               #  the users passwords!
               # detailperm = 0600
        # }

        # Create a unique accounting session Id.  Many NASes re-use or
        # repeat values for Acct-Session-Id, causing no end of
        # confusion.
        #
        #  This module will add a (probably) unique session id
        #  to an accounting packet based on the attributes listed
        #  below found in the packet.  See doc/rlm_acct_unique for
        #  more information.
        #
        acct_unique {
               key = "User-Name, Acct-Session-Id, NAS-IP-Address, Client-IP-Address, NAS-
Port"
        }


        #  Include another file that has the SQL-related configuration.
        #  This is another file only because it tends to be big.
        #
        #  The following configuration file is for use with MySQL.
        #
        #  For Postgresql, use:               ${confdir}/postgresql.conf
        #  For MS-SQL, use:           ${confdir}/mssql.conf
        #  For Oracle, use:           ${confdir}/oraclesql.conf
        #
        $INCLUDE  ${confdir}/sql.conf


        #  For Cisco VoIP specific accounting with Postgresql,
        #  use:        ${confdir}/pgsql-voip.conf
        #
        #  You will also need the sql schema from:
        #        src/billing/cisco_h323_db_schema-postgres.sql
        #  Note: This config can be use AS WELL AS the standard sql
        #  config if you need SQL based Auth


        #  Write a 'utmp' style file, of which users are currently
        #  logged in, and where they've logged in from.
        #
        #  This file is used mainly for Simultaneous-Use checking,
        #  and also 'radwho', to see who's currently logged in.
        #
        radutmp {
               #  Where the file is stored.  It's not a log file,
               #  so it doesn't need rotating.
               #
               filename = ${logdir}/radutmp

               #  The field in the packet to key on for the
               #  'user' name,  If you have other fields which you want
               #  to use to key on to control Simultaneous-Use,
               #  then you can use them here.
               #
               #  Note, however, that the size of the field in the
               #  'utmp' data structure is small, around 32
               #  characters, so that will limit the possible choices
               #  of keys.
               #
               #  You may want instead: %{Stripped-User-Name:-%{User-Name}}
               username = %{User-Name}


               #  Whether or not we want to treat "user" the same
               #  as "USER", or "User".  Some systems have problems
               #  with case sensitivity, so this should be set to
               #  'no' to enable the comparisons of the key attribute
               #  to be case insensitive.
               #
               case_sensitive = yes

               #  Accounting information may be lost, so the user MAY
               #  have logged off of the NAS, but we haven't noticed.
               #  If so, we can verify this information with the NAS,
               #
               #  If we want to believe the 'utmp' file, then this
               #  configuration entry can be set to 'no'.
               #
               check_with_nas = yes

               # Set the file permissions, as the contents of this file
               # are usually private.
               perm = 0600

               callerid = "yes"
        }

        # "Safe" radutmp - does not contain caller ID, so it can be
        # world-readable, and radwho can work for normal users, without
        # exposing any information that isn't already exposed by who(1).
        #
        # This is another 'instance' of the radutmp module, but it is given
        # then name "sradutmp" to identify it later in the "accounting"
        # section.
        radutmp sradutmp {
               filename = ${logdir}/sradutmp
               perm = 0644
               callerid = "no"
        }

        # attr_filter - filters the attributes received in replies from
        # proxied servers, to make sure we send back to our RADIUS client
        # only allowed attributes.
        attr_filter {
               attrsfile = ${confdir}/attrs
        }

        #  counter module:
        #  This module takes an attribute (count-attribute).
        #  It also takes a key, and creates a counter for each unique
        #  key.  The count is incremented when accounting packets are
        #  received by the server.  The value of the increment depends
        #  on the attribute type.
        #  If the attribute is Acct-Session-Time or of an integer type we add the
        #  value of the attribute. If it is anything else we increase the
        #  counter by one.
        #
        #  The 'reset' parameter defines when the counters are all reset to
        #  zero.  It can be hourly, daily, weekly, monthly or never.
        #
        #  hourly: Reset on 00:00 of every hour
        #  daily: Reset on 00:00:00 every day
        #  weekly: Reset on 00:00:00 on sunday
        #  monthly: Reset on 00:00:00 of the first day of each month
        #
        #  It can also be user defined. It should be of the form:
        #  num[hdwm] where:
        #  h: hours, d: days, w: weeks, m: months
        #  If the letter is ommited days will be assumed. In example:
        #  reset = 10h (reset every 10 hours)
        #  reset = 12  (reset every 12 days)
        #
        #
        #  The check-name attribute defines an attribute which will be
        #  registered by the counter module and can be used to set the
        #  maximum allowed value for the counter after which the user
        #  is rejected.
        #  Something like:
        #
        #  DEFAULT Max-Daily-Session := 36000
        #          Fall-Through = 1
        #
        #  You should add the counter module in the instantiate
        #  section so that it registers check-name before the files
        #  module reads the users file.
        #
        #  If check-name is set and the user is to be rejected then we
        #  send back a Reply-Message and we log a Failure-Message in
        #  the radius.log
        #  If the count attribute is Acct-Session-Time then on each login
        #  we send back the remaining online time as a Session-Timeout attribute
        #
        #  The counter-name can also be used instead of using the check-name
        #  like below:
        #
        #  DEFAULT  Daily-Session-Time > 3600, Auth-Type = Reject
        #      Reply-Message = "You've used up more than one hour today"
        #
        #  The allowed-servicetype attribute can be used to only take
        #  into account specific sessions. For example if a user first
        #  logs in through a login menu and then selects ppp there will
        #  be two sessions. One for Login-User and one for Framed-User
        #  service type. We only need to take into account the second one.
        #
        #  The module should be added in the instantiate, authorize and
        #  accounting sections.  Make sure that in the authorize
        #  section it comes after any module which sets the
        #  'check-name' attribute.
        #
        counter daily {
                filename = ${raddbdir}/db.daily
                key = User-Name
                count-attribute = Acct-Session-Time
                reset = daily
                counter-name = Daily-Session-Time
                check-name = Max-Daily-Session
                allowed-servicetype = Framed-User
                cache-size = 5000
        }

        # The "always" module is here for debugging purposes. Each
        # instance simply returns the same result, always, without
        # doing anything.
        always fail {
                rcode = fail
        }
        always reject {
                rcode = reject
        }
```

```
always ok {
        rcode = ok
        simulcount = 0
        mpp = no
}

#
#  The 'expression' module currently has no configuration.
#
#  This module is useful only for 'xlat'.  To use it,
#  put 'exec' into the 'instantiate' section.  You can then
#  do dynamic translation of attributes like:
#
#  Attribute-Name = `%{expr:2 + 3 + %{exec: uid -u}}`
#
#  The value of the attribute will be replaced with the output
#  of the program which is executed.  Due to RADIUS protocol
#  limitations, any output over 253 bytes will be ignored.
expr {
}

#
#  The 'digest' module currently has no configuration.
#
#  "Digest" authentication against a Cisco SIP server.
#  See 'doc/rfc/draft-sterman-aaa-sip-00.txt' for details
#  on performing digest authentication for Cisco SIP servers.
#
digest {
}

#
#  Execute external programs
#
#  This module is useful only for 'xlat'.  To use it,
#  put 'exec' into the 'instantiate' section.  You can then
#  do dynamic translation of attributes like:
#
#  Attribute-Name = `%{exec:/path/to/program args}`
#
#  The value of the attribute will be replaced with the output
#  of the program which is executed.  Due to RADIUS protocol
#  limitations, any output over 253 bytes will be ignored.
#
#  The RADIUS attributes from the user request will be placed
#  into environment variables of the executed program, as
#  described in 'doc/variables.txt'
#
exec {
        wait = yes
        input_pairs = request
}

#
#  This is a more general example of the execute module.
#
#  This one is called "echo".
#
#  Attribute-Name = `%{echo:/path/to/program args}`
#
#  If you wish to execute an external program in more than
#  one section (e.g. 'authorize', 'pre_proxy', etc), then it
#  is probably best to define a different instance of the
#  'exec' module for every section.
#
exec echo {
        #
        #  Wait for the program to finish.
        #
        #  If we do NOT wait, then the program is "fire and
        #  forget", and any output attributes from it are ignored.
        #
        #  If we are looking for the program to output
        #  attributes, and want to add those attributes to the
        #  request, then we MUST wait for the program to
        #  finish, and therefore set 'wait=yes'
        #
        # allowed values: {no, yes}
        wait = yes

        #
        #  The name of the program to execute, and it's
        #  arguments.  Dynamic translation is done on this
        #  field, so things like the following example will
        #  work.
        #
        program = "/bin/echo %{User-Name}"

        #
        #  The attributes which are placed into the
        #  environment variables for the program.
        #
        #  Allowed values are:
        #
        #       request        attributes from the request
        #       config         attributes from the configuration items list
        #       reply          attributes from the reply
        #       proxy-request  attributes from the proxy request
        #       proxy-reply    attributes from the proxy reply
        #
        #  Note that some attributes may not exist at some
        #  stages.  e.g. There may be no proxy-reply
        #  attributes if this module is used in the
        #  'authorize' section.
        #
        input_pairs = request

        #
        #  Where to place the output attributes (if any) from
        #  the executed program.  The values allowed, and the
        #  restrictions as to availability, are the same as
        #  for the input_pairs.
        #
        output_pairs = reply

        #
        #  When to execute the program.  If the packet
        #  type does NOT match what's listed here, then
        #  the module does NOT execute the program.
        #
        #  For a list of allowed packet types, see
        #  the 'dictionary' file, and look for VALUEs
        #  of the Packet-Type attribute.
        #
        #  By default, the module executes on ANY packet.
        #  Un-comment out the following line to tell the
        #  module to execute only if an Access-Accept is
        #  being sent to the NAS.
        #
        #packet_type = Access-Accept
}

#  Do server side ip pool management. Should be added in post-auth and
#  accounting sections.
#
#  The module also requires the existance of the Pool-Name
#  attribute. That way the administrator can add the Pool-Name
#  attribute in the user profiles and use different pools
#  for different users. The Pool-Name attribute is a *check* item not
#  a reply item.
#
# Example:
        # radiusd.conf: ippool students { [...] }
        # users file  : DEFAULT Group == students, Pool-Name := "students"
        #
        # ********* IF YOU CHANGE THE RANGE PARAMETERS YOU MUST *********
        # ********* THEN ERASE THE DB FILES                     *********
        #
        ippool main_pool {

                #  range-start,range-stop: The start and end ip
                #  addresses for the ip pool
                range-start = 192.168.1.1
                range-stop = 192.168.3.254

                #  netmask: The network mask used for the ip's
                netmask = 255.255.255.0

                #  cache-size: The gdbm cache size for the db
                #  files. Should be equal to the number of ip's
                #  available in the ip pool
                cache-size = 800

                # session-db: The main db file used to allocate ip's to clients
                session-db = ${raddbdir}/db.ippool

                # ip-index: Helper db index file used in multilink
                ip-index = ${raddbdir}/db.ipindex

                # override: Will this ippool override a Framed-IP-Address already set
                override = no

                # maximum-timeout: If not zero specifies the maximum time in seconds an
                # entry may be active. Default: 0
                maximum-timeout = 0
        }

        # ANSI X9.9 token support.  Not included by default.
        # $INCLUDE  ${confdir}/x99.conf

}

# Instantiation
#
#  This section orders the loading of the modules.  Modules
#  listed here will get loaded BEFORE the later sections like
#  authorize, authenticate, etc. get examined.
#
#  This section is not strictly needed.  When a section like
#  authorize refers to a module, it's automatically loaded and
#  initialized.  However, some modules may not be listed in any
#  of the following sections, so they can be listed here.
#
#  Also, listing modules here ensures that you have control over
#  the order in which they are initalized.  If one module needs
#  something defined by another module, you can list them in order
#  here, and ensure that the configuration will be OK.
#
instantiate {
        #
        #  Allows the execution of external scripts.
        #  The entire command line (and output) must fit into 253 bytes.
        #
        #  e.g. Framed-Pool = `%{exec:/bin/echo foo}`
        exec

        #
        #  The expression module doesn't do authorization,
        #  authentication, or accounting.  It only does dynamic
        #  translation, of the form:
        #
        #       Session-Timeout = `%{expr:2 + 3}`
        #
        #  So the module needs to be instantiated, but CANNOT be
        #  listed in any other section.  See 'doc/rlm_expr' for
        #  more information.
        #
        expr

        #
        # We add the counter module here so that it registers
        # the check-name attribute before any module which sets
        # it
#       daily
}

#  Authorization. First preprocess (hints and huntgroups files),
#  then realms, and finally look in the "users" file.
#
#  The order of the realm modules will determine the order that
#  we try to find a matching realm.
#
#  Make *sure* that 'preprocess' comes before any realm if you
#  need to setup hints for the remote radius server
authorize {
        #
        #  The preprocess module takes care of sanitizing some bizarre
        #  attributes in the request, and turning them into attributes
        #  which are more standard.
        #
        #  It takes care of processing the 'raddb/hints' and the
        #  'raddb/huntgroups' files.
        #
        #  It also adds the %{Client-IP-Address} attribute to the request.
        preprocess

        #
        #  If you want to have a log of authentication requests,
        #  un-comment the following line, and the 'detail auth_log'
        #  section, above.
#       auth_log

#       attr_filter

        #
        #  The chap module will set 'Auth-Type := CHAP' if we are
        #  handling a CHAP request and Auth-Type has not already been set
        chap

        #
        #  If the users are logging in with an MS-CHAP-Challenge
        #  attribute for authentication, the mschap module will find
        #  the MS-CHAP-Challenge attribute, and add 'Auth-Type := MS-CHAP'
        #  to the request, which will cause the server to then use
        #  the mschap module for authentication.
        mschap

        #
        #  If you have a Cisco SIP server authenticating against
        #  FreeRADIUS, uncomment the following line, and the 'digest'
        #  line in the 'authenticate' section.
        digest

        #
        #  Look for IPASS style 'realm/', and if not found, look for
        #  '@realm', and decide whether or not to proxy, based on
        #  that.
#       IPASS

        #
        #  If you are using multiple kinds of realms, you probably
        #  want to set "ignore_null = yes" for all of them.
        #  Otherwise, when the first style of realm doesn't match,
        #  the other styles won't be checked.
        #
        suffix
#       ntdomain

        #
        #  This module takes care of EAP-MD5, EAP-TLS, and EAP-LEAP
        #  authentication.
        #
        #  It also sets the EAP-Type attribute in the request
        #  attribute list to the EAP type from the packet.
        eap

        #
        #  Read the 'users' file
        files

        #
        #  Look in an SQL database.  The schema of the database
        #  is meant to mirror the "users" file.
        #
        #  See "Authorization Queries" in sql.conf
#       sql

        #
        #  If you are using /etc/smbpasswd, and are also doing
        #  mschap authentication, the un-comment this line, and
        #  configure the 'etc_smbpasswd' module, above.
#       etc_smbpasswd

        #
        #  The ldap module will set Auth-Type to LDAP if it has not
        #  already been set
#       ldap

        #
        #  Enforce daily limits on time spent logged in.
#       daily

        #
        # Use the checkval module
#       checkval
}


#  Authentication.
#
#
#  This section lists which modules are available for authentication.
#  Note that it does NOT mean 'try each module in order'.  It means
#  that a module from the 'authorize' section adds a configuration
#  attribute 'Auth-Type := FOO'.  That authentication type is then
#  used to pick the apropriate module from the list below.
#

#  In general, you SHOULD NOT set the Auth-Type attribute.  The server
#  will figure it out on its own, and will do the right thing.  The
#  most common side effect of erroneously setting the Auth-Type
#  attribute is that one authentication method will work, but the
#  others will not.
#
#  The common reasons to set the Auth-Type attribute by hand
#  is to either forcibly reject the user, or forcibly accept him.
#
authenticate {
        #
        #  PAP authentication, when a back-end database listed
        #  in the 'authorize' section supplies a password.  The
        #  password can be clear-text, or encrypted.
        Auth-Type PAP {
                pap
        }
        #
        #  Most people want CHAP authentication
        #  A back-end database listed in the 'authorize' section
        #  MUST supply a CLEAR TEXT password.  Encrypted passwords
        #  won't work.
        Auth-Type CHAP {
                chap
        }

        #
        #  MSCHAP authentication.
        Auth-Type MS-CHAP {
                mschap
        }

        #
        #  If you have a Cisco SIP server authenticating against
        #  FreeRADIUS, uncomment the following line, and the 'digest'
        #  line in the 'authorize' section.
#       digest

        #
        #  Pluggable Authentication Modules.
#       pam

        #
        #  See 'man getpwent' for information on how the 'unix'
        #  module checks the users password.  Note that packets
        #  containing CHAP-Password attributes CANNOT be authenticated
        #  against /etc/passwd!  See the FAQ for details.
        #
        unix

        # Uncomment it if you want to use ldap for authentication
        #
        # Note that this means "check plain-text password against
        # the ldap database", which means that EAP won't work,
        # as it does not supply a plain-text password.
#       Auth-Type LDAP {
#               ldap
#       }
#
        #
        #  Allow EAP authentication.
        eap
}


#
#  Pre-accounting.  Decide which accounting type to use.
#
preacct {
        preprocess

        #
        #  Ensure that we have a semi-unique identifier for every
        #  request, and many NAS boxes are broken.
        acct_unique

        #
        #  Look for IPASS-style 'realm/', and if not found, look for
        #  '@realm', and decide whether or not to proxy, based on
        #  that.
        #
        #  Accounting requests are generally proxied to the same
        #  home server as authentication requests.
#       IPASS
        suffix
#       ntdomain
```

```
        #
        #  Read the 'acct_users' file
        files
}

#
#  Accounting.  Log the accounting data.
#
accounting {
        #
        #  Create a 'detail'ed log of the packets.
        #  Note that accounting requests which are proxied
        #  are also logged in the detail file.
        detail
#       daily

        #  Update the wtmp file
        #
        #  If you don't use "radlast", you can delete this line.
        unix

        #
        #  For Simultaneous-Use tracking.
        #
        #  Due to packet losses in the network, the data here
        #  may be incorrect.  There is little we can do about it.
        radutmp
#       sradutmp

        #  Return an address to the IP Pool when we see a stop record.
#       main_pool

        #
        #  Log traffic to an SQL database.
        #
        #  See "Accounting queries" in sql.conf
#       sql


        #  Cisco VoIP specific bulk accounting
#       pgsql-voip

}


#  Session database, used for checking Simultaneous-Use. Either the radutmp
#  or rlm_sql module can handle this.
#  The rlm_sql module is *much* faster
session {
        radutmp

        #
        #  See "Simultaneous Use Checking Querie" in sql.conf
#       sql
}


#  Post-Authentication
#  Once we KNOW that the user has been authenticated, there are
#  additional steps we can take.
post-auth {
        #  Get an address from the IP Pool.
#       main_pool

        #
        #  If you want to have a log of authentication replies,
        #  un-comment the following line, and the 'detail reply_log'
        #  section, above.
#       reply_log

        #
        #  After authenticating the user, do another SQL qeury.
        #
        #  See "Authentication Logging Queries" in sql.conf
#       sql

        #
        #  Access-Reject packets are sent through the REJECT sub-section
        #  of the post-auth section.
        #
#       Post-Auth-Type REJECT {
#               insert-module-name-here
#       }

}

#
#  When the server decides to proxy a request to a home server,
#  the proxied request is first passed through the pre-proxy
#  stage.  This stage can re-write the request, or decide to
#  cancel the proxy.
#
#  Only a few modules currently have this method.
#
pre-proxy {
#       attr_rewrite

        #  If you want to have a log of packets proxied to a home
        #  server, un-comment the following line, and the
        #  'detail pre_proxy_log' section, above.
#       pre_proxy_log
}

#
#  When the server receives a reply to a request it proxied
#  to a home server, the request may be massaged here, in the
#  post-proxy stage.
#
post-proxy {
        #

        #  If you want to have a log of replies from a home server,
        #  un-comment the following line, and the 'detail post_proxy_log'
        #  section, above.
#       post_proxy_log

#       attr_rewrite

        #  Uncomment the following line if you want to filter replies from
        #  remote proxies based on the rules defined in the 'attrs' file.

#       attr_filter

        #
        #  If you are proxying LEAP, you MUST configure the EAP
        #  module, and you MUST list it here, in the post-proxy
        #  stage.
        #
        #  You MUST also use the 'nostrip' option in the 'realm'
        #  configuration.  Otherwise, the User-Name attribute
        #  in the proxied request will not match the user name
        #  hidden inside of the EAP packet, and the end server will
        #  reject the EAP request.
        #
        eap
}
```

users:

```
DEFAULT EAP-Type == EAP-TLS, Auth-Type := Reject
DEFAULT Auth-Type = System
        Fall-Through = 1
```