APPENDIX

All Appendix,Installer, and All Manuals are included in Source CD.

TABLE OF APPENDIX

APPENDIX A

```
[root@localhost delta]# radiusd -X
FreeRADIUS Version 2.1.8, for host i686-pc-linux-gnu, built on Jan 13 2010 at 21:47:21
Starting - reading configuration files ...
including configuration file /usr/local/etc/raddb/radiusd.conf
including configuration file /usr/local/etc/raddb/proxy.conf
including configuration file /usr/local/etc/raddb/clients.conf
including files in directory /usr/local/etc/raddb/modules/
including configuration file /usr/local/etc/raddb/modules/passwd
including configuration file /usr/local/etc/raddb/modules/mac2ip
including configuration file /usr/local/etc/raddb/modules/ntlm_auth
including configuration file /usr/local/etc/raddb/modules/preprocess
including configuration file /usr/local/etc/raddb/modules/exec
including configuration file /usr/local/etc/raddb/modules/unix
including configuration file /usr/local/etc/raddb/modules/etc_group
including configuration file /usr/local/etc/raddb/modules/radutmp
including configuration file /usr/local/etc/raddb/modules/ippool
including configuration file /usr/local/etc/raddb/modules/realm
including configuration file /usr/local/etc/raddb/modules/files
including configuration file /usr/local/etc/raddb/modules/counter
including configuration file /usr/local/etc/raddb/modules/attr_rewrite
including configuration file /usr/local/etc/raddb/modules/expr
including configuration file /usr/local/etc/raddb/modules/cui
including configuration file /usr/local/etc/raddb/modules/ldap
including configuration file /usr/local/etc/raddb/modules/perl
including configuration file /usr/local/etc/raddb/modules/krb5
including configuration file /usr/local/etc/raddb/modules/logintime
including configuration file /usr/local/etc/raddb/modules/attr_filter
including configuration file /usr/local/etc/raddb/modules/mac2vlan
including configuration file /usr/local/etc/raddb/modules/checkval
including configuration file /usr/local/etc/raddb/modules/policy
including configuration file /usr/local/etc/raddb/modules/chap
including configuration file /usr/local/etc/raddb/modules/detail.example.com
including configuration file /usr/local/etc/raddb/modules/pam
including configuration file /usr/local/etc/raddb/modules/mschap
including configuration file /usr/local/etc/raddb/modules/acct_unique
including configuration file /usr/local/etc/raddb/modules/wimax
including configuration file /usr/local/etc/raddb/modules/sradutmp
including configuration file /usr/local/etc/raddb/modules/sql_log
including configuration file /usr/local/etc/raddb/modules/expiration
including configuration file /usr/local/etc/raddb/modules/sqlcounter_expire_on_login
including configuration file /usr/local/etc/raddb/eap.conf
including configuration file /usr/local/etc/raddb/sql.conf
including configuration file /usr/local/etc/raddb/sql/mysql/dialup.conf
```

```
including configuration file /usr/local/etc/raddb/policy.conf
including files in directory /usr/local/etc/raddb/sites-enabled/
including configuration file /usr/local/etc/raddb/sites-enabled/control-socket
including configuration file /usr/local/etc/raddb/sites-enabled/default
including configuration file /usr/local/etc/raddb/sites-enabled/inner-tunnel
main {
        allow_core_dumps = no
}
including dictionary file /usr/local/etc/raddb/dictionary
main {
        prefix = "/usr/local"
        localstatedir = "/usr/local/var"
        logdir = "/usr/local/var/log/radius"
        libdir = "/usr/local/lib"
        radacctdir = "/usr/local/var/log/radius/radacct"
        hostname_lookups = no
        max_request_time = 30
        cleanup_delay = 5
        max_requests = 1024
        pidfile = "/usr/local/var/run/radiusd/radiusd.pid"
        checkrad = "/usr/local/sbin/checkrad"
        debug_level = 0
        proxy_requests = no
 log {
        stripped_names = no
        auth = yes
        auth_badpass = yes
        auth_goodpass = yes
 }
 security {
        max_attributes = 200
        reject_delay = 1
        status_server = yes
 }
}
radiusd: #### Loading Realms and Home Servers ####
 proxy server {
        retry_delay = 5
        retry_count = 3
        default_fallback = no
        dead_time = 120
        wake_all_if_all_dead = no
 }
 home_server localhost {
        ipaddr = 127.0.0.1
        port = 1812
        type = "auth"
```

```
        secret = "testing123"
        response_window = 20
        max_outstanding = 65536
        require_message_authenticator = no
        zombie_period = 40
        status_check = "status-server"
        ping_interval = 30
        check_interval = 30
        num_answers_to_alive = 3
        num_pings_to_alive = 3
        revive_interval = 120
        status_check_timeout = 4
        irt = 2
        mrt = 16
        mrc = 5
        mrd = 30
 }
 home_server_pool my_auth_failover {
        type = fail-over
        home_server = localhost
 }
 realm example.com {
        auth_pool = my_auth_failover
 }
 realm LOCAL {
 }
radiusd: #### Loading Clients ####
 client localhost {
        ipaddr = 127.0.0.1
        require_message_authenticator = no
        secret = "rahasia"
        shortname = "localhost"
        nastype = "other"
 }
 client 192.168.0.0/24 {
        require_message_authenticator = no
        secret = "rahasia1"
        shortname = "private-network-1"
 }
 client 192.168.0.0/16 {
        require_message_authenticator = no
        secret = "rahasia2"
        shortname = "private-network-2"
 }
radiusd: #### Instantiating modules ####
 instantiate {
 Module: Linked to module rlm_exec
```

```
 Module: Instantiating exec
  exec {
       wait = no
       input_pairs = "request"
       shell_escape = yes
  }
 Module: Linked to module rlm_expr
 Module: Instantiating expr
 Module: Linked to module rlm_expiration
 Module: Instantiating expiration
  expiration {
       reply-message = "Password Has Expired  "
  }
 Module: Linked to module rlm_logintime
 Module: Instantiating logintime
  logintime {
       reply-message = "You are calling outside your allowed timespan  "
       minimum-timeout = 60
  }
 }
radiusd: #### Loading Virtual Servers ####
server inner-tunnel {
 modules {
 Module: Checking authenticate {...} for more modules to load
 Module: Linked to module rlm_pap
 Module: Instantiating pap
  pap {
       encryption_scheme = "auto"
       auto_header = no
  }
 Module: Linked to module rlm_chap
 Module: Instantiating chap
 Module: Linked to module rlm_mschap
 Module: Instantiating mschap
  mschap {
       use_mppe = yes
       require_encryption = no
       require_strong = no
       with_ntdomain_hack = no
  }
 Module: Linked to module rlm_unix
 Module: Instantiating unix
  unix {
       radwtmp = "/usr/local/var/log/radius/radwtmp"
  }
 Module: Linked to module rlm_eap
 Module: Instantiating eap
```

```
 eap {
      default_eap_type = "tls"
      timer_expire = 60
      ignore_unknown_eap_types = no
      cisco_accounting_username_bug = no
      max_sessions = 4096
 }
Module: Linked to sub-module rlm_eap_gtc
Module: Instantiating eap-gtc
  gtc {
      challenge = "Password: "
      auth_type = "PAP"
  }
Module: Linked to sub-module rlm_eap_tls
Module: Instantiating eap-tls
  tls {
      rsa_key_exchange = no
      dh_key_exchange = yes
      rsa_key_length = 512
      dh_key_length = 512
      verify_depth = 0
      pem_file_type = yes
      private_key_file = "/usr/local/etc/raddb/certs/server.pem"
      certificate_file = "/usr/local/etc/raddb/certs/server.pem"
      CA_file = "/usr/local/etc/raddb/certs/ca.pem"
      private_key_password = "rahasia"
      dh_file = "/usr/local/etc/raddb/certs/dh"
      random_file = "/usr/local/etc/raddb/certs/random"
      fragment_size = 1024
      include_length = yes
      check_crl = no
      cipher_list = "DEFAULT"
      make_cert_command = "/usr/local/etc/raddb/certs/bootstrap"
   cache {
      enable = no
      lifetime = 24
      max_entries = 255
   }
  }
Module: Linked to sub-module rlm_eap_ttls
Module: Instantiating eap-ttls
  ttls {
      default_eap_type = "md5"
      copy_request_to_tunnel = no
      use_tunneled_reply = no
      virtual_server = "inner-tunnel"
      include_length = yes
```

```
 }
Module: Linked to sub-module rlm_eap_peap
Module: Instantiating eap-peap
 peap {
      default_eap_type = "mschapv2"
      copy_request_to_tunnel = no
      use_tunneled_reply = no
      proxy_tunneled_request_as_eap = yes
      virtual_server = "inner-tunnel"
 }
Module: Linked to sub-module rlm_eap_mschapv2
Module: Instantiating eap-mschapv2
 mschapv2 {
      with_ntdomain_hack = no
 }
Module: Checking authorize {...} for more modules to load
Module: Linked to module rlm_realm
Module: Instantiating suffix
 realm suffix {
      format = "suffix"
      delimiter = "@"
      ignore_default = no
      ignore_null = no
 }
Module: Linked to module rlm_files
Module: Instantiating files
 files {
      usersfile = "/usr/local/etc/raddb/users"
      acctusersfile = "/usr/local/etc/raddb/acct_users"
      preproxy_usersfile = "/usr/local/etc/raddb/preproxy_users"
      compat = "no"
 }
Module: Checking session {...} for more modules to load
Module: Linked to module rlm_radutmp
Module: Instantiating radutmp
 radutmp {
      filename = "/usr/local/var/log/radius/radutmp"
      username = "%{User-Name}"
      case_sensitive = yes
      check_with_nas = yes
      perm = 384
      callerid = yes
 }
Module: Checking post-proxy {...} for more modules to load
Module: Checking post-auth {...} for more modules to load
Module: Linked to module rlm_attr_filter
Module: Instantiating attr_filter.access_reject
```

```
 attr_filter attr_filter.access_reject {
        attrsfile = "/usr/local/etc/raddb/attrs.access_reject"
        key = "%{User-Name}"
 }
 } # modules
} # server
server {
 modules {
 Module: Checking authenticate {...} for more modules to load
 Module: Checking authorize {...} for more modules to load
 Module: Linked to module rlm_preprocess
 Module: Instantiating preprocess
 preprocess {
        huntgroups = "/usr/local/etc/raddb/huntgroups"
        hints = "/usr/local/etc/raddb/hints"
        with_ascend_hack = no
        ascend_channels_per_line = 23
        with_ntdomain_hack = no
        with_specialix_jetstream_hack = no
        with_cisco_vsa_hack = no
        with_alvarion_vsa_hack = no
 }
 Module: Linked to module rlm_detail
 Module: Instantiating auth_log
 detail auth_log {
        detailfile = "/usr/local/var/log/radius/radacct/%{Client-IP-Address}/auth-detail-%Y%m
%d"
        header = "%t"
        detailperm = 384
        dirperm = 493
        locking = no
        log_packet_header = no
 }
 Module: Checking preacct {...} for more modules to load
 Module: Linked to module rlm_acct_unique
 Module: Instantiating acct_unique
 acct_unique {
        key = "User-Name, Acct-Session-Id, NAS-IP-Address, Client-IP-Address, NAS-Port"
 }
 Module: Checking accounting {...} for more modules to load
 Module: Instantiating detail
 detail {
        detailfile = "/usr/local/var/log/radius/radacct/%{Client-IP-Address}/detail-%Y%m%d"
        header = "%t"
        detailperm = 384
        dirperm = 493
        locking = no
```

```
        log_packet_header = no
 }
 Module: Instantiating attr_filter.accounting_response
 attr_filter attr_filter.accounting_response {
        attrsfile = "/usr/local/etc/raddb/attrs.accounting_response"
        key = "%{User-Name}"
 }
 Module: Checking session {...} for more modules to load
 Module: Checking post-proxy {...} for more modules to load
 Module: Checking post-auth {...} for more modules to load
 } # modules
} # server
radiusd: #### Opening IP addresses and Ports ####
listen {
        type = "auth"
        ipaddr = *
        port = 0
}
listen {
        type = "acct"
        ipaddr = *
        port = 0
}
listen {
        type = "control"
 listen {
        socket = "/usr/local/var/run/radiusd/radiusd.sock"
 }
}
Listening on authentication address * port 1812
Listening on accounting address * port 1813
Listening on command file /usr/local/var/run/radiusd/radiusd.sock
Ready to process requests.
```

APPENDIX B

```
## radiusd.conf -- FreeRADIUS server configuration file.
##
##      http://www.freeradius.org/
##      $Id$
##

#######################################################################
#
#       Read "man radiusd" before editing this file.  See the section
#       titled DEBUGGING.  It outlines a method where you can quickly
#       obtain the configuration you want, without running into
#       trouble.
#
#       Run the server in debugging mode, and READ the output.
#
#               $ radiusd -X
#
#       We cannot emphasize this point strongly enough.  The vast
#       majority of problems can be solved by carefully reading the
#       debugging output, which includes warnings about common issues,
#       and suggestions for how they may be fixed.
#
#       There may be a lot of output, but look carefully for words like:
#       "warning", "error", "reject", or "failure".  The messages there
#       will usually be enough to guide you to a solution.
#
#       If you are going to ask a question on the mailing list, then
#       explain what you are trying to do, and include the output from
#       debugging mode (radiusd -X).  Failure to do so means that all
#       of the responses to your question will be people telling you
#       to "post the output of radiusd -X".

#######################################################################
#
#       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.
#
#       See "man radiusd.conf" for documentation on the format of this
#       file.  Note that the individual configuration items are NOT
#       documented in that "man" page.  They are only documented here,
#       in the comments.
#
```

```
#       As of 2.0.0, FreeRADIUS supports a simple processing language
#       in the "authorize", "authenticate", "accounting", etc. sections.
#       See "man unlang" for details.
#

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

#
#  name of the running server.  See also the "-n" command-line option.
name = radiusd

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

# Should likely be ${localstatedir}/lib/radiusd
db_dir = ${raddbdir}

#
# 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 = ${exec_prefix}/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 = ${run_dir}/${name}.pid

#  chroot: directory where the server does "chroot".
#
#  The chroot is done very early in the process of starting the server.
#  After the chroot has been performed it switches to the "user" listed
#  below (which MUST be specified).  If "group" is specified, it switchs
#  to that group, too.  Any other groups listed for the specified "user"
#  in "/etc/group" are also added as part of this process.
#
#  The current working directory (chdir / cd) is left *outside* of the
#  chroot until all of the modules have been initialized.  This allows
#  the "raddb" directory to be left outside of the chroot.  Once the
#  modules have been initialized, it does a "chdir" to ${logdir}.  This
#  means that it should be impossible to break out of the chroot.
#
#  If you are worried about security issues related to this use of chdir,
#  then simply ensure that the "raddb" directory is inside of the chroot,
#  end be sure to do "cd raddb" BEFORE starting the server.
#
#  If the server is statically linked, then the only files that have
#  to exist in the chroot are ${run_dir} and ${logdir}.  If you do the
#  "cd raddb" as discussed above, then the "raddb" directory has to be
#  inside of the chroot directory, too.
#
#chroot = /path/to/chroot/directory

# 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 radius'.
#
#  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.
#
#  The server will also try to use "initgroups" to read /etc/groups.
#  It will join all groups where "user" is a member.  This can allow
#  for some finer-grained access controls.
#
#user = radius
#group = radius

#  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

#  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

#  listen: Make the server listen on a particular IP address, and send
#  replies out from that address. This directive is most useful for
#  hosts with multiple IP addresses on one interface.
#
#  If you want the server to listen on additional addresses, or on
#  additionnal ports, you can use multiple "listen" sections.
#
#  Each section make the server listen for only one type of packet,
#  therefore authentication and accounting have to be configured in
#  different sections.
#
#  The server ignore all "listen" section if you are using '-i' and '-p'
#  on the command line.
#
listen {
        #  Type of packets to listen for.
        #  Allowed values are:
        #       auth    listen for authentication packets
        #       acct    listen for accounting packets
        #       proxy   IP to use for sending proxied packets
        #       detail  Read from the detail file.  For examples, see
        #         raddb/sites-available/copy-acct-to-home-server
        #       status  listen for Status-Server packets.  For examples,
        #                 see raddb/sites-available/status
        #       coa     listen for CoA-Request and Disconnect-Request
        #                 packets.  For examples, see the file
        #                 raddb/sites-available/coa-server
```

```
        #
        type = auth

        #  Note: "type = proxy" lets you control the source IP used for
        #       proxying packets, with some limitations:
        #
        #    * A proxy listener CANNOT be used in a virtual server section.
        #    * You should probably set "port = 0".
        #    * Any "clients" configuration will be ignored.
        #
        #  See also proxy.conf, and the "src_ipaddr" configuration entry
        #  in the sample "home_server" section.  When you specify the
        #  source IP address for packets sent to a home server, the
        #  proxy listeners are automatically created.

        #  IP address on which to listen.
        #  Allowed values are:
        #          dotted quad (1.2.3.4)
        #      hostname    (radius.example.com)
        #      wildcard    (*)
        ipaddr = *
#
#       testing
#       ipaddr = 192.168.1.10

        #  OR, you can use an IPv6 address, but not both
        #  at the same time.
#       ipv6addr = ::     # any.  ::1 == localhost

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

        #  Some systems support binding to an interface, in addition
        #  to the IP address.  This feature isn't strictly necessary,
        #  but for sites with many IP addresses on one interface,
        #  it's useful to say "listen on all addresses for eth0".
        #
        #  If your system does not support this feature, you will
        #  get an error if you try to use it.
        #
#       interface = eth0

        #  Per-socket lists of clients.  This is a very useful feature.
        #
        #  The name here is a reference to a section elsewhere in
        #  radiusd.conf, or clients.conf.  Having the name as
        #  a reference allows multiple sockets to use the same
        #  set of clients.
```

```
        #
        #  If this configuration is used, then the global list of clients
        #  is IGNORED for this "listen" section.  Take care configuring
        #  this feature, to ensure you don't accidentally disable a
        #  client you need.
        #
        #  See clients.conf for the configuration of "per_socket_clients".
        #
#       clients = per_socket_clients
}

#  This second "listen" section is for listening on the accounting
#  port, too.
#
listen {
        ipaddr = *
#       ipv6addr = ::
        port = 0
        type = acct
#       interface = eth0
#       clients = per_socket_clients
}

#  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

#
#  Logging section.  The various "log_*" configuration items
#  will eventually be moved here.
#
log {
        #
        #  Destination for log messages.  This can be one of:
        #
        #       files - log to "file", as defined below.
        #       syslog - to syslog (see also the "syslog_facility", below.
        #       stdout - standard output
        #       stderr - standard error.
        #
        #  The command-line option "-X" over-rides this option, and forces
        #  logging to go to stdout.
        #
        destination = files

        #
        #  The logging messages for the server are appended to the
        #  tail of this file if destination == "files"
        #
        #  If the server is running in debugging mode, this file is
        #  NOT used.
        #
        file = ${logdir}/radius.log

        #
        #  If this configuration parameter is set, then log messages for
        #  a *request* go to this file, rather than to radius.log.
        #
        #  i.e. This is a log file per request, once the server has accepted
        #  the request as being from a valid client.  Messages that are
        #  not associated with a request still go to radius.log.
        #
        #  Not all log messages in the server core have been updated to use
        #  this new internal API.  As a result, some messages will still
        #  go to radius.log.  Please submit patches to fix this behavior.
        #
        #  The file name is expanded dynamically.  You should ONLY user
        #  server-side attributes for the filename (e.g. things you control).
        #  Using this feature MAY also slow down the server substantially,
        #  especially if you do thinks like SQL calls as part of the
```

```
#  expansion of the filename.
#
#  The name of the log file should use attributes that don't change
#  over the lifetime of a request, such as User-Name,
#  Virtual-Server or Packet-Src-IP-Address.  Otherwise, the log
#  messages will be distributed over multiple files.
#
#  Logging can be enabled for an individual request by a special
#  dynamic expansion macro:  %{debug: 1}, where the debug level
#  for this request is set to '1' (or 2, 3, etc.).  e.g.
#
#          ...
#          update control {
#               Tmp-String-0 = "%{debug:1}"
#          }
#          ...
#
#  The attribute that the value is assigned to is unimportant,
#  and should be a "throw-away" attribute with no side effects.
#
#requests = ${logdir}/radiusd-%{%{Virtual-Server}:-DEFAULT}-%Y%m%d.log

#
#  Which syslog facility to use, if ${destination} == "syslog"
#
#  The exact values permitted here are OS-dependent.  You probably
#  don't want to change this.
#
syslog_facility = daemon

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

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

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

```
        #  Log additional text at the end of the "Login OK" messages.
        #  for these to work, the "auth" and "auth_goopass" or "auth_badpass"
        #  configurations above have to be set to "yes".
        #
        #  The strings below are dynamically expanded, which means that
        #  you can put anything you want in them.  However, note that
        #  this expansion can be slow, and can negatively impact server
        #  performance.
        #
#       msg_goodpass = ""
#       msg_badpass = ""
}

#  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

        #
        #  reject_delay: 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.
        #
        #  When sent a Status-Server message, the server responds with
        #  an Access-Accept or Accounting-Response packet.
        #
        #  This is mainly useful for administrators who want to "ping"
        #  the server, without adding test users, or creating fake
        #  accounting packets.
        #
        #  It's also useful when a NAS marks a RADIUS server "dead".
        #  The NAS can periodically "ping" the server with a Status-Server
        #  packet.  If the server responds, it must be alive, and the
        #  NAS can start using it for real requests.
        #
        #  See also raddb/sites-available/status
        #
        status_server = yes
}

# 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  = no
$INCLUDE 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 clients.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
        #  for an example.
        #

        #
        #  As of 2.0.5, most of the module configurations are in a
        #  sub-directory.  Files matching the regex /[a-zA-Z0-9_.]+/
        #  are loaded.  The modules are initialized ONLY if they are
        #  referenced in a processing section, such as authorize,
        #  authenticate, accounting, pre/post-proxy, etc.
        #
        $INCLUDE ${confdir}/modules/

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

        #  Include another file that has the SQL-related configuration.
        #  This is another file only because it tends to be big.
        #
        $INCLUDE sql.conf

        #
        #  This module is an SQL enabled version of the counter module.
        #
        #  Rather than maintaining seperate (GDBM) databases of
        #  accounting info for each counter, this module uses the data
        #  stored in the raddacct table by the sql modules. This
        #  module NEVER does any database INSERTs or UPDATEs.  It is
        #  totally dependent on the SQL module to process Accounting
        #  packets.
        #
#       $INCLUDE sql/mysql/counter.conf

        #
        #  IP addresses managed in an SQL table.
        #
#       $INCLUDE sqlippool.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
        expiration
        logintime

        # subsections here can be thought of as "virtual" modules.
        #
        # e.g. If you have two redundant SQL servers, and you want to
        # use them in the authorize and accounting sections, you could
        # place a "redundant" block in each section, containing the
        # exact same text.  Or, you could uncomment the following
        # lines, and list "redundant_sql" in the authorize and
        # accounting sections.
        #
        #redundant redundant_sql {
        #       sql1
        #       sql2
```

```
        #}
}

$INCLUDE policy.conf

######################################################################
#
#       Load virtual servers.
#
#       This next $INCLUDE line loads files in the directory that
#       match the regular expression: /[a-zA-Z0-9_.]+/
#
#       It allows you to define new virtual servers simply by placing
#       a file into the raddb/sites-enabled/ directory.
#
$INCLUDE sites-enabled/

######################################################################
#
#       All of the other configuration sections like "authorize {}",
#       "authenticate {}", "accounting {}", have been moved to the
#       the file:
#
#               raddb/sites-available/default
#
#       This is the "default" virtual server that has the same
#       configuration as in version 1.0.x and 1.1.x.  The default
#       installation enables this virtual server.  You should
#       edit it to create policies for your local site.
#
#       For more documentation on virtual servers, see:
#
#               raddb/sites-available/README
#
```

APPENDIX C

```
# -*- text -*-
##
## clients.conf -- client configuration directives
##
##	$Id$

#######################################################################
#
#  Define RADIUS clients (usually a NAS, Access Point, etc.).

#
#  Defines a RADIUS client.
#
#  '127.0.0.1' is another name for 'localhost'.  It is enabled by default,
#  to allow testing of the server after an initial installation.  If you
#  are not going to be permitting RADIUS queries from localhost, we suggest
#  that you delete, or comment out, this entry.
#
#

#
#  Each client has a "short name" that is used to distinguish it from
#  other clients.
#
#  In version 1.x, the string after the word "client" was the IP
#  address of the client.  In 2.0, the IP address is configured via
#  the "ipaddr" or "ipv6addr" fields.  For compatibility, the 1.x
#  format is still accepted.
#
client localhost {
	#  Allowed values are:
	#	dotted quad (1.2.3.4)
	#	hostname    (radius.example.com)
	ipaddr = 127.0.0.1

	#  OR, you can use an IPv6 address, but not both
	#  at the same time.
#	ipv6addr = ::	# any.  ::1 == localhost

	#
	#  A note on DNS:  We STRONGLY recommend using IP addresses
	#  rather than host names.  Using host names means that the
	#  server will do DNS lookups when it starts, making it
	#  dependent on DNS.  i.e. If anything goes wrong with DNS,
	#  the server won't start!
	#
	#  The server also looks up the IP address from DNS once, and
```

```
#  only once, when it starts.  If the DNS record is later
#  updated, the server WILL NOT see that update.
#

#  One client definition can be applied to an entire network.
#  e.g. 127/8 should be defined with "ipaddr = 127.0.0.0" and
#  "netmask = 8"
#
#  If not specified, the default netmask is 32 (i.e. /32)
#
#  We do NOT recommend using anything other than 32.  There
#  are usually other, better ways to acheive the same goal.
#  Using netmasks of other than 32 can cause security issues.
#
#  You can specify overlapping networks (127/8 and 127.0/16)
#  In that case, the smallest possible network will be used
#  as the "best match" for the client.
#
#  Clients can also be defined dynamically at run time, based
#  on any criteria.  e.g. SQL lookups, keying off of NAS-Identifier,
#  etc.
#  See raddb/sites-available/dynamic-clients for details.
#

#       netmask = 32

#
#  The shared secret use to "encrypt" and "sign" packets between
#  the NAS and FreeRADIUS.  You MUST change this secret from the
#  default, otherwise it's not a secret any more!
#
#  The secret can be any string, up to 8k characters in length.
#
#  Control codes can be entered vi octal encoding,
#       e.g. "\101\102" == "AB"
#  Quotation marks can be entered by escaping them,
#       e.g. "foo\"bar"
#
#  A note on security:  The security of the RADIUS protocol
#  depends COMPLETELY on this secret!  We recommend using a
#  shared secret that is composed of:
#
#       upper case letters
#       lower case letters
#       numbers
#
#  And is at LEAST 8 characters long, preferably 16 characters in
#  length.  The secret MUST be random, and should not be words,
#  phrase, or anything else that is recognizable.
#
#  The default secret below is only for testing, and should
```

```
#  not be used in any real environment.
#
secret          = rahasia

#
#  Old-style clients do not send a Message-Authenticator
#  in an Access-Request.  RFC 5080 suggests that all clients
#  SHOULD include it in an Access-Request.  The configuration
#  item below allows the server to require it.  If a client
#  is required to include a Message-Authenticator and it does
#  not, then the packet will be silently discarded.
#
#  allowed values: yes, no
require_message_authenticator = no

#
#  The short name is used as an alias for the fully qualified
#  domain name, or the IP address.
#
#  It is accepted for compatibility with 1.x, but it is no
#  longer necessary in 2.0
#
shortname       = localhost

#
# the following three fields are optional, but may be used by
# checkrad.pl for simultaneous use checks
#

#
# The nastype tells 'checkrad.pl' which NAS-specific method to
#  use to query the NAS for simultaneous use.
#
#  Permitted NAS types are:
#
#       cisco
#       computone
#       livingston
#       max40xx
#       multitech
#       netserver
#       pathras
#       patton
#       portslave
#       tc
#       usrhiper
#       other           # for all other types

#
nastype     = other     # localhost isn't usually a NAS...
```

```
        #
        #  The following two configurations are for future use.
        #  The 'naspasswd' file is currently used to store the NAS
        #  login name and password, which is used by checkrad.pl
        #  when querying the NAS for simultaneous use.
        #
#       login      = !root
#       password   = someadminpas

        #
        #  As of 2.0, clients can also be tied to a virtual server.
        #  This is done by setting the "virtual_server" configuration
        #  item, as in the example below.
        #
#       virtual_server = home1

        #
        #  A pointer to the "home_server_pool" OR a "home_server"
        #  section that contains the CoA configuration for this
        #  client.  For an example of a coa home server or pool,
        #  see raddb/sites-available/originate-coa
#       coa_server = coa
}

# IPv6 Client
#client ::1 {
#       secret          = testing123
#       shortname       = localhost
#}
#
# All IPv6 Site-local clients
#client fe80::/16 {
#       secret          = testing123
#       shortname       = localhost
#}

#client some.host.org {
#       secret          = testing123
#       shortname       = localhost
#}

#
#  You can now specify one secret for a network of clients.
#  When a client request comes in, the BEST match is chosen.
#  i.e. The entry from the smallest possible network.
#
client 192.168.0.0/24 {
        secret          = rahasia1
        shortname       = private-network-1
}
```

```
client 192.168.0.0/16 {
        secret          = rahasia2
        shortname       = private-network-2
}
#client 10.10.10.10 {
#       # secret and password are mapped through the "secrets" file.
#       secret     = testing123
#       shortname   = liv1
#     # the following three fields are optional, but may be used by
#     # checkrad.pl for simultaneous usage checks
#       nastype     = livingston
#       login      = !root
#       password    = someadminpas
#}

#############################################################################
#
#  Per-socket client lists.  The configuration entries are exactly
#  the same as above, but they are nested inside of a section.
#
#  You can have as many per-socket client lists as you have "listen"
#  sections, or you can re-use a list among multiple "listen" sections.
#
#  Un-comment this section, and edit a "listen" section to add:
#  "clients = per_socket_clients".  That IP address/port combination
#  will then accept ONLY the clients listed in this section.
#
#clients per_socket_clients {
#       client 192.168.3.4 {
#               secret = testing123
#     }
#}
```

APPENDIX D

```
#
#       Please read the documentation file ../doc/processing_users_file,
#       or 'man 5 users' (after installing the server) for more information.
#
#       This file contains authentication security and configuration
#       information for each user.  Accounting requests are NOT processed
#       through this file.  Instead, see 'acct_users', in this directory.
#
#       If you are not sure why a particular reply is being sent by the
#       server, then run the server in debugging mode (radiusd -X), and
#       you will see which entries in this file are matched.
#
#       When an authentication request is received from the comm server,
#       these values are tested. Only the first match is used unless the
#       "Fall-Through" variable is set to "Yes".
#
#       A special user named "DEFAULT" matches on all usernames.
#       You can have several DEFAULT entries. All entries are processed
#       in the order they appear in this file. The first entry that
#       matches the login-request will stop processing unless you use
#       the Fall-Through variable.
#
#       If you use the database support to turn this file into a .db or .dbm
#       file, the DEFAULT entries _have_ to be at the end of this file and
#       you can't have multiple entries for one username.
#
#       Indented (with the tab character) lines following the first
#       line indicate the configuration values to be passed back to
#       the comm server to allow the initiation of a user session.
#       This can include things like the PPP configuration values
#       or the host to log the user onto.
#
#       You can include another `users' file with `$INCLUDE users.other'
#
#
#       For a list of RADIUS attributes, and links to their definitions,
#       see:
#
#       http://www.freeradius.org/rfc/attributes.html
#

#
# Deny access for a specific user.  Note that this entry MUST
# be before any other 'Auth-Type' attribute which results in the user
# being authenticated.
#
# Note that there is NO 'Fall-Through' attribute, so the user will not
# be given any additional resources.
```

```
#
#lameuser       Auth-Type := Reject
#               Reply-Message = "Your account has been disabled."

#
# Deny access for a group of users.
#
# Note that there is NO 'Fall-Through' attribute, so the user will not
# be given any additional resources.
#
#DEFAULT        Group == "disabled", Auth-Type := Reject
#               Reply-Message = "Your account has been disabled."
#

#
# This is a complete entry for "steve". Note that there is no Fall-Through
# entry so that no DEFAULT entry will be used, and the user will NOT
# get any attributes in addition to the ones listed here.
#
#steve  Cleartext-Password := "testing"
#       Service-Type = Framed-User,
#       Framed-Protocol = PPP,
#       Framed-IP-Address = 172.16.3.33,
#       Framed-IP-Netmask = 255.255.255.0,
#       Framed-Routing = Broadcast-Listen,
#       Framed-Filter-Id = "std.ppp",
#       Framed-MTU = 1500,
#       Framed-Compression = Van-Jacobsen-TCP-IP

#
# This is an entry for a user with a space in their name.
# Note the double quotes surrounding the name.
#
#"John Doe"     Cleartext-Password := "hello"
#               Reply-Message = "Hello, %{User-Name}"

#
# Dial user back and telnet to the default host for that port
#
#Deg    Cleartext-Password := "ge55ged"
#       Service-Type = Callback-Login-User,
#       Login-IP-Host = 0.0.0.0,
#       Callback-Number = "9,5551212",
#       Login-Service = Telnet,
#       Login-TCP-Port = Telnet

#
# Another complete entry. After the user "dialbk" has logged in, the
# connection will be broken and the user will be dialed back after which
# he will get a connection to the host "timeshare1".
#
```

```
#dialbk Cleartext-Password := "callme"
#       Service-Type = Callback-Login-User,
#       Login-IP-Host = timeshare1,
#       Login-Service = PortMaster,
#       Callback-Number = "9,1-800-555-1212"

#
# user "swilson" will only get a static IP number if he logs in with
# a framed protocol on a terminal server in Alphen (see the huntgroups file).
#
# Note that by setting "Fall-Through", other attributes will be added from
# the following DEFAULT entries
#
#swilson        Service-Type == Framed-User, Huntgroup-Name == "alphen"
#               Framed-IP-Address = 192.168.1.65,
#               Fall-Through = Yes

#
# If the user logs in as 'username.shell', then authenticate them
# using the default method, give them shell access, and stop processing
# the rest of the file.
#
#DEFAULT        Suffix == ".shell"
#               Service-Type = Login-User,
#               Login-Service = Telnet,
#               Login-IP-Host = your.shell.machine


#
# The rest of this file contains the several DEFAULT entries.
# DEFAULT entries match with all login names.
# Note that DEFAULT entries can also Fall-Through (see first entry).
# A name-value pair from a DEFAULT entry will _NEVER_ override
# an already existing name-value pair.
#

#
# Set up different IP address pools for the terminal servers.
# Note that the "+" behind the IP address means that this is the "base"
# IP address. The Port-Id (S0, S1 etc) will be added to it.
#
#DEFAULT        Service-Type == Framed-User, Huntgroup-Name == "alphen"
#               Framed-IP-Address = 192.168.1.32+,
#               Fall-Through = Yes

#DEFAULT        Service-Type == Framed-User, Huntgroup-Name == "delft"
#               Framed-IP-Address = 192.168.2.32+,
#               Fall-Through = Yes

# Sample defaults for all framed connections.
#
```

```
#DEFAULT        Service-Type == Framed-User
#               Framed-IP-Address = 255.255.255.254,
#               Framed-MTU = 576,
#               Service-Type = Framed-User,
#               Fall-Through = Yes

#
# Default for PPP: dynamic IP address, PPP mode, VJ-compression.
# NOTE: we do not use Hint = "PPP", since PPP might also be auto-detected
#       by the terminal server in which case there may not be a "P" suffix.
#       The terminal server sends "Framed-Protocol = PPP" for auto PPP.
#
DEFAULT         Framed-Protocol == PPP
        Framed-Protocol = PPP,
        Framed-Compression = Van-Jacobson-TCP-IP

#
# Default for CSLIP: dynamic IP address, SLIP mode, VJ-compression.
#
DEFAULT         Hint == "CSLIP"
        Framed-Protocol = SLIP,
        Framed-Compression = Van-Jacobson-TCP-IP

#
# Default for SLIP: dynamic IP address, SLIP mode.
#
DEFAULT         Hint == "SLIP"
        Framed-Protocol = SLIP

#
# Last default: rlogin to our main server.
#
#DEFAULT
#       Service-Type = Login-User,
#       Login-Service = Rlogin,
#       Login-IP-Host = shellbox.ispdomain.com

# #
# # Last default: shell on the local terminal server.
# #
# DEFAULT
#       Service-Type = Administrative-User

# On no match, the user is denied access.
#
delta   Cleartext-Password = "delta"
```

APPENDIX E

```
##########################################################################
#
#       As of 2.0.0, FreeRADIUS supports virtual hosts using the
#       "server" section, and configuration directives.
#
#       Virtual hosts should be put into the "sites-available"
#       directory.  Soft links should be created in the "sites-enabled"
#       directory to these files.  This is done in a normal installation.
#
#       $Id$
#
##########################################################################
#
#       Read "man radiusd" before editing this file.  See the section
#       titled DEBUGGING.  It outlines a method where you can quickly
#       obtain the configuration you want, without running into
#       trouble.  See also "man unlang", which documents the format
#       of this file.
#
#       This configuration is designed to work in the widest possible
#       set of circumstances, with the widest possible number of
#       authentication methods.  This means that in general, you should
#       need to make very few changes to this file.
#
#       The best way to configure the server for your local system
#       is to CAREFULLY edit this file.  Most attempts to make large
#       edits to this file will BREAK THE SERVER.  Any edits should
#       be small, and tested by running the server with "radiusd -X".
#       Once the edits have been verified to work, save a copy of these
#       configuration files somewhere.  (e.g. as a "tar" file).  Then,
#       make more edits, and test, as above.
#
#       There are many "commented out" references to modules such
#       as ldap, sql, etc.  These references serve as place-holders.
#       If you need the functionality of that module, then configure
#       it in radiusd.conf, and un-comment the references to it in
#       this file.  In most cases, those small changes will result
#       in the server being able to connect to the DB, and to
#       authenticate users.
#
##########################################################################

#
#       In 1.x, the "authorize", etc. sections were global in
#       radiusd.conf.  As of 2.0, they SHOULD be in a server section.
#
```

```
#       The server section with no virtual server name is the "default"
#       section.  It is used when no server name is specified.
#
#       We don't indent the rest of this file, because doing so
#       would make it harder to read.
#

#  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.
        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

        #
        #  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

        #
```

```
        #  The WiMAX specification says that the Calling-Station-Id
        #  is 6 octets of the MAC.  This definition conflicts with
        #  RFC 3580, and all common RADIUS practices.  Un-commenting
        #  the "wimax" module here means that it will fix the
        #  Calling-Station-Id attribute to the normal format as
        #  specified in RFC 3580 Section 3.21
#       wimax

        #
        #  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.
        #
        #  As of 2.0, the EAP module returns "ok" in the authorize stage
        #  for TTLS and PEAP.  In 1.x, it never returned "ok" here, so
        #  this change is compatible with older configurations.
        #
        #  The example below uses module failover to avoid querying all
        #  of the following modules if the EAP module returns "ok".
        #  Therefore, your LDAP and/or SQL servers will not be queried
        #  for the many packets that go back and forth to set up TTLS
        #  or PEAP.  The load on those servers will therefore be reduced.
        #
        eap {
                ok = return
        }

        #
        #  Pull crypt'd passwords from /etc/passwd or /etc/shadow,
        #  using the system API's to get the password.  If you want
        #  to read /etc/passwd or /etc/shadow directly, see the
        #  passwd module in radiusd.conf.
        #
        unix
```

```
        #
        #  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

        expiration
        logintime

        #
        #  If no other module has claimed responsibility for
        #  authentication, then try to use PAP.  This allows the
        #  other modules listed above to add a "known good" password
        #  to the request, and to do nothing else.  The PAP module
        #  will then see that password, and use it to do PAP
        #  authentication.
        #
        #  This module should be listed last, so that the other modules
        #  get a chance to set Auth-Type for themselves.
        #
#       pap

        #
        #  If "status_server = yes", then Status-Server messages are passed
        #  through the following section, and ONLY the following section.
        #  This permits you to do DB queries, for example.  If the modules
```

```
        #  listed here return "fail", then NO response is sent.
        #
#       Autz-Type Status-Server {
#
#       }
}


#  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 (Auth-Type := Reject),
#  or to or forcibly accept the user (Auth-Type := Accept).
#
#  Note that Auth-Type := Accept will NOT work with EAP.
#
#  Please do not put "unlang" configurations into the "authenticate"
#  section.  Put them in the "post-auth" section instead.  That's what
#  the post-auth section is for.
#
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

        #
        #  The older configurations sent a number of attributes in
        #  Access-Challenge packets, which wasn't strictly correct.
        #  If you want to filter out these attributes, uncomment
        #  the following lines.
        #
#       Auth-Type eap {
#               eap {
#                       handled = 1
#               }
#               if (handled && (Response-Packet-Type == Access-Challenge)) {
#                       attr_filter.access_challenge.post-auth
#                       handled  # override the "updated" code from attr_filter
```

```
#                       }
#               }
}


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

        #
        #  Session start times are *implied* in RADIUS.
        #  The NAS never sends a "start time".  Instead, it sends
        #  a start packet, *possibly* with an Acct-Delay-Time.
        #  The server is supposed to conclude that the start time
        #  was "Acct-Delay-Time" seconds in the past.
        #
        #  The code below creates an explicit start time, which can
        #  then be used in other modules.
        #
        #  The start time is: NOW - delay - session_length
        #

#         update request {
#                 FreeRADIUS-Acct-Session-Start-Time = "%{expr: %l - %{%{Acct-Session-Time}:-0} - %
{%{Acct-Delay-Time}:-0}}"
#       }


        #
        #  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

        #
        #  If you receive stop packets with zero session length,
        #  they will NOT be logged in the database.  The SQL module
        #  will print a message (only in debugging mode), and will
        #  return "noop".
        #
        #  You can ignore these packets by uncommenting the following
        #  three lines.  Otherwise, the server will not respond to the
        #  accounting request, and the NAS will retransmit.
        #
#       if (noop) {
#               ok
#       }

        #
        #  Instead of sending the query to the SQL server,
        #  write it into a log file.
        #
```

```
#       sql_log

        # Cisco VoIP specific bulk accounting
#       pgsql-voip

        # Filter attributes from the accounting response.
        attr_filter.accounting_response

        #
        # See "Autz-Type Status-Server" for how this works.
        #
#       Acct-Type Status-Server {
#
#       }
}


# 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 Queries" 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 query.
        #
        # See "Authentication Logging Queries" in sql.conf
#       sql

        #
        # Instead of sending the query to the SQL server,
        # write it into a log file.
```

```
        #
#       sql_log

        #
        #  Un-comment the following if you have set
        #  'edir_account_policy_check = yes' in the ldap module sub-section of
        #  the 'modules' section.
        #
#       ldap

        exec

        #
        #  Calculate the various WiMAX keys.  In order for this to work,
        #  you will need to define the WiMAX NAI, usually via
        #
        #       update request {
        #           WiMAX-MN-NAI = "%{User-Name}"
        #       }
        #
        #  If you want various keys to be calculated, you will need to
        #  update the reply with "template" values.  The module will see
        #  this, and replace the template values with the correct ones
        #  taken from the cryptographic calculations.  e.g.
        #
        #       update reply {
        #               WiMAX-FA-RK-Key = 0x00
        #               WiMAX-MSK = "%{EAP-MSK}"
        #       }
        #
        #  You may want to delete the MS-MPPE-*-Keys from the reply,
        #  as some WiMAX clients behave badly when those attributes
        #  are included.  See "raddb/modules/wimax", configuration
        #  entry "delete_mppe_keys" for more information.
        #
#       wimax

        #  If the WiMAX module did it's work, you may want to do more
        #  things here, like delete the MS-MPPE-*-Key attributes.
        #
        #       if (updated) {
        #               update reply {
        #                       MS-MPPE-Recv-Key !* 0x00
        #                       MS-MPPE-Send-Key !* 0x00
        #               }
        #       }

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

```
        #  Add the ldap module name (or instance) if you have set
        #  'edir_account_policy_check = yes' in the ldap module configuration
        #
        Post-Auth-Type REJECT {
                attr_filter.access_reject
        }
}

#
#  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

        #  Uncomment the following line if you want to change attributes
        #  as defined in the preproxy_users file.
#       files

        #  Uncomment the following line if you want to filter requests
        #  sent to remote servers based on the rules defined in the
        #  'attrs.pre-proxy' file.
#       attr_filter.pre-proxy

        #  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.post-proxy
```

```
        #
        #  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

        #
        #  If the server tries to proxy a request and fails, then the
        #  request is processed through the modules in this section.
        #
        #  The main use of this section is to permit robust proxying
        #  of accounting packets.  The server can be configured to
        #  proxy accounting packets as part of normal processing.
        #  Then, if the home server goes down, accounting packets can
        #  be logged to a local "detail" file, for processing with
        #  radrelay.  When the home server comes back up, radrelay
        #  will read the detail file, and send the packets to the
        #  home server.
        #
        #  With this configuration, the server always responds to
        #  Accounting-Requests from the NAS, but only writes
        #  accounting packets to disk if the home server is down.
        #
#       Post-Proxy-Type Fail {
#                       detail
#       }

}
```

APPENDIX F

```
# -*- text -*-
##
##  eap.conf -- Configuration for EAP types (PEAP, TTLS, etc.)
##
##      $Id$

#####################################################################
#
#  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.
#
#  EAP types NOT listed here may be supported via the "eap2" module.
#  See experimental.conf for documentation.
#
        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 = tls

                #  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

                #
                # Help prevent DoS attacks by limiting the number of
                # sessions that the server is tracking.  Most systems
                # can handle ~30 EAP sessions/s, so the default limit
                # of 4096 should be OK.
                max_sessions = 4096

                # 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
#
#  See raddb/certs/README for additional comments
#  on certificates.
#
#  If OpenSSL was not found at the time the server was
#  built, the "tls", "ttls", and "peap" sections will
#  be ignored.
#
#  Otherwise, when the server first starts in debugging
#  mode, test certificates will be created.  See the
#  "make_cert_command" below for details, and the README
#  file in raddb/certs
#
#  These test certificates SHOULD NOT be used in a normal
#  deployment.  They are created only to make it easier
#  to install the server, and to perform some simple
#  tests with EAP-TLS, TTLS, or PEAP.
#
#  See also:
#
#  http://www.dslreports.com/forum/remark,9286052~mode=flat
#
tls {
        #
        #  These is used to simplify later configurations.
```

```
#
certdir = ${confdir}/certs
cadir = ${confdir}/certs

private_key_password = rahasia
private_key_file = ${certdir}/server.pem

#  If Private key & Certificate are located in
#  the same file, then private_key_file &
#  certificate_file must contain the same file
#  name.
#
#  If CA_file (below) is not used, then the
#  certificate_file below MUST include not
#  only the server certificate, but ALSO all
#  of the CA certificates used to sign the
#  server certificate.
certificate_file = ${certdir}/server.pem

#  Trusted Root CA list
#
#  ALL of the CA's in this list will be trusted
#  to issue client certificates for authentication.
#
#  In general, you should use self-signed
#  certificates for 802.1x (EAP) authentication.
#  In that case, this CA file should contain
#  *one* CA certificate.
#
#  This parameter is used only for EAP-TLS,
#  when you issue client certificates.  If you do
#  not use client certificates, and you do not want
#  to permit EAP-TLS authentication, then delete
#  this configuration item.
CA_file = ${cadir}/ca.pem

#
#  For DH cipher suites to work, you have to
#  run OpenSSL to create the DH file first:
#
#       openssl dhparam -out certs/dh 1024
#
dh_file = ${certdir}/dh
random_file = ${certdir}/random

#
#  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) uncomment the line below.
        # 5) Restart radiusd
#       check_crl = yes
#       CA_path = /path/to/directory/with/ca_certs/and/crls/

    #
    # If check_cert_issuer is set, the value will
    # be checked against the DN of the issuer in
    # the client certificate.  If the values do not
    # match, the cerficate verification will fail,
    # rejecting the user.
    #
#     check_cert_issuer = "/C=GB/ST=Berkshire/L=Newbury/O=My Company Ltd"

    #
    # 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.
    #
    # This check is done only if the previous
    # "check_cert_issuer" is not set, or if
    # the check succeeds.
    #
#       check_cert_cn = %{User-Name}
#
        # Set this option to specify the allowed
        # TLS cipher suites.  The format is listed
        # in "man 1 ciphers".
        cipher_list = "DEFAULT"
```

```
        #

        #  This configuration entry should be deleted
        #  once the server is running in a normal
        #  configuration.  It is here ONLY to make
        #  initial deployments easier.
        #
        make_cert_command = "${certdir}/bootstrap"

        #
        #  Session resumption / fast reauthentication
        #  cache.
        #
        cache {
            #
            #  Enable it.  The default is "no".
            #  Deleting the entire "cache" subsection
            #  Also disables caching.
            #
            #  You can disallow resumption for a
            #  particular user by adding the following
            #  attribute to the control item list:
            #
            #              Allow-Session-Resumption = No
            #
            #  If "enable = no" below, you CANNOT
            #  enable resumption for just one user
            #  by setting the above attribute to "yes".
            #
            enable = no

            #
            #  Lifetime of the cached entries, in hours.
            #  The sessions will be deleted after this
            #  time.
            #
            lifetime = 24 # hours

            #
            #  The maximum number of entries in the
            #  cache.  Set to "0" for "infinite".
            #
            #  This could be set to the number of users
            #  who are logged in... which can be a LOT.
            #
            max_entries = 255
        }
}

#  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.
#
#  You can make TTLS require a client cert by setting
#
#       EAP-TLS-Require-Client-Cert = Yes
#
#  in the control items for a request.
#
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 inner tunneled request can be sent
        #  through a virtual server constructed
        #  specifically for this purpose.
        #
        #  If this entry is commented out, the inner
        #  tunneled request will be sent through
        #  the virtual server that processed the
        #  outer requests.
        #
        virtual_server = "inner-tunnel"

        #  This has the same meaning as the
        #  same field in the "tls" module, above.
        #  The default value here is "yes".
#       include_length = yes
}

######################################################
#
#  !!!!! WARNINGS for Windows compatibility  !!!!!
#
######################################################
#
#  If you see the server send an Access-Challenge,
#  and the client never sends another Access-Request,
#  then
#
#                  STOP!
#
#  The server certificate has to have special OID's
#  in it, or else the Microsoft clients will silently
#  fail.  See the "scripts/xpextensions" file for
#  details, and the following page:
#
#       http://support.microsoft.com/kb/814394/en-us
#
#  For additional Windows XP SP2 issues, see:
#
#       http://support.microsoft.com/kb/885453/en-us
#
#  Note that we do not necessarily agree with their
#  explanation... but the fix does appear to work.
```

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

#
#  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.
#
#
#  You can make PEAP require a client cert by setting
#
#       EAP-TLS-Require-Client-Cert = Yes
#
#  in the control items for a request.
#
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

        #  the PEAP module also has these configuration
        #  items, which are the same as for TTLS.
        copy_request_to_tunnel = no
        use_tunneled_reply = no

        #  When the tunneled session is proxied, the
        #  home server may not understand EAP-MSCHAP-V2.
        #  Set this entry to "no" to proxy the tunneled
        #  EAP-MSCHAP-V2 as normal MSCHAPv2.
#       proxy_tunneled_request_as_eap = yes

        #
        #  The inner tunneled request can be sent
        #  through a virtual server constructed
        #  specifically for this purpose.
        #
        #  If this entry is commented out, the inner
```

```
        #  tunneled request will be sent through
        #  the virtual server that processed the
        #  outer requests.
        #
        virtual_server = "inner-tunnel"
}

#
#  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 {
}
}
```

APPENDIX G

```
######################################################################
#  Configuration for the SQL module
#  The database schemas and queries are located in subdirectories:
#
#       sql/DB/schema.sql       Schema
#       sql/DB/dialup.conf      Basic dialup (including policy) queries
#       sql/DB/counter.conf     counter
#       sql/DB/ippool.conf      IP Pools in SQL
#       sql/DB/ippool.sql       schema for IP pools.

#
sql {
        #
        #  Set the database to one of:
        #
        #       mysql, mssql, oracle, postgresql
        #
        database = "mysql"

        #
        #  Which FreeRADIUS driver to use.
        #
        driver = "rlm_sql_${database}"
        #driver = "rlm_sql-2.1.8.so"

        # Connection info:
        server = "localhost"
        port = 3306
        login = "radius"
        password = "radius"

        # Database table configuration for everything except Oracle
        radius_db = "radius"
        # If you are using Oracle then use this instead
         #  radius_db  =  "(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=localhost)(PORT=1521))
(CONNECT_DATA=(SID=your_sid)))"

        # If you want both stop and start records logged to the
        # same SQL table, leave this as is.  If you want them in
        # different tables, put the start table in acct_table1
        # and stop table in acct_table2
        acct_table1 = "radacct"
        acct_table2 = "radacct"

        # Allow for storing data after authentication
        postauth_table = "radpostauth"
```

```
        authcheck_table = "radcheck"
        authreply_table = "radreply"

        groupcheck_table = "radgroupcheck"
        groupreply_table = "radgroupreply"

        # Table to keep group info
        usergroup_table = "usergroup"

        # If set to 'yes' (default) we read the group tables
        # If set to 'no' the user MUST have Fall-Through = Yes in the radreply table
         read_groups = yes

        # Remove stale session if checkrad does not see a double login
        deletestalesessions = yes

        # Print all SQL statements when in debug mode (-x)
        sqltrace = yes
        sqltracefile = ${logdir}/sqltrace.sql

        # number of sql connections to make to server
        num_sql_socks = 5

        # number of seconds to dely retrying on a failed database
        # connection (per_socket)
        connect_failure_retry_delay = 60

        # closed "lifetime" seconds after they were first opened.
        lifetime = 0

        # Maximum number of queries used by an SQL socket.  If you are
        # limit the number of queries performed over one socket.  After
        # "max_qeuries", the socket will be closed.  Use 0 for "no limit".
        max_queries = 0

        # Set to 'yes' to read radius clients from the database ('nas' table)
        # Clients will ONLY be read on server startup.  For performance
        # and security reasons, finding clients via SQL queries CANNOT
        # be done "live" while the server is running.
        #
        #readclients = yes

        # Table to keep radius client info
        nas_table = "nas"

        # Read driver-specific configuration
        $INCLUDE sql/${database}/dialup.conf
}
```

APPENDIX H

```
#
# Main Configuration File
#
# it can be default or whatever language. Only greek are supported
# from non latin alphabet languages
# These attribute only apply for ldap not for sql
#
general_prefered_lang: el
general_prefered_lang_name: Greek
#
# Uncomment this if normal attributes (not the ;lang-xx ones) in ldap
# are utf8 encoded.
#
#general_decode_normal_attributes: yes
#
general_base_dir: /usr/local/dialup_admin
general_radiusd_base_dir: /usr/local/radiusd
general_domain: company.com
#
# Set it to yes to use sessions and cache the various mappings
# You can also set use_session = 1 in config.php3 to also cache
# the admin.conf
#
# ---- IMPORTANT -- IMPORTANT -- IMPORTANT ----
#Remember to use the 'Clear Cache' page if you use sessions and do any changes
#in any of the configuration files.
#
general_use_session: no
#
# This is used by the failed logins page. It states the default back time
# in minutes.
#
general_most_recent_fl: 30

#
# Realm setup
#
# Set general_strip_realms to yes in order  to stip realms from usernames.
# By default realms are not striped
#general_strip_realms : yes
#
# The delimiter used  in realms. Default is @
#
general_realm_delimiter: @
#
# The format of the realms. Can be either suffix (realm is after the username)
# or prefix (realm is before the username). Default is suffix
```

```
#
general_realm_format: suffix
#


general_ldap_attrmap: %{general_radiusd_base_dir}/etc/raddb/ldap.attrmap
general_sql_attrmap: %{general_base_dir}/conf/sql.attrmap
general_extra_ldap_attrmap: %{general_base_dir}/conf/extra.ldap-attrmap
#
# it can be either ldap or sql
#
general_lib_type: sql
general_user_edit_attrs_file: %{general_base_dir}/conf/user_edit.attrs
general_sql_attrs_file: %{general_base_dir}/conf/sql.attrs
general_default_file: %{general_base_dir}/conf/default.vals
#general_ld_library_path: /usr/local/snmpd/lib
#
# can be 'snmp' (for snmpfinger) or empty to query the radacct table without first
# querying the nas
#
general_finger_type: snmp
general_snmpfinger_bin: %{general_base_dir}/bin/snmpfinger
general_radclient_bin: %{general_radiusd_base_dir}/bin/radclient
#
# this information is used from the server check page
#
general_test_account_login: test
general_test_account_password: testpass
#
# These are used as default values for the user test page
#
general_radius_server: localhost
general_radius_server_port: 1812
#
# can be either pap or chap
#
general_radius_server_auth_proto: pap
#
# sorry, single valued for now. Should become something like
# password[server-name]: xxxxx
#
general_radius_server_secret: XXXXXX
general_auth_request_file: %{general_base_dir}/conf/auth.request
#
# can be one of crypt,md5,clear
#
general_encryption_method: crypt
#
# can be either asc (older dates first) or desc (recent dates first)
#
general_accounting_info_order: desc
```

```
nas1_name: nas1.%{general_domain}
nas1_model: Cisco 2511 access server
nas1_ip: 147.122.122.121
nas1_port_num: 16
nas1_community: public
nas2_name: nas2.%{general_domain}
nas2_model: Cisco 2511 access server
nas2_ip: 147.122.122.123
nas2_port_num: 16
nas2_community: public
nas3_name: nas3.%{general_domain}
nas3_model: Cisco 5300 access server
nas3_ip: 147.122.122.124
nas3_port_num: 210
nas3_community: public

#
# The ldap server to connect to.
# Both ldap_server and ldap_write_server can be a space-separated
# list of ldap hostnames. In that case the library will try to connect
# to the servers in the order that they appear. If the first host is down
# ldap_connect will ask for the second ldap host and so on.
#
ldap_server: ldap.%{general_domain}
#
# There are many cases where we have a small write master and
# a lot of fast read only replicas. If that is the case uncomment
# ldap_write_server and point it to the write master. It will be
# used only when writing to the directory, not when reading
#
#ldap_write_server: master.%{general_domain}
ldap_base: dc=company,dc=com
ldap_binddn: cn=Directory Manager
ldap_bindpw: XXXXXXX
ldap_default_new_entry_suffix: ou=dialup,ou=guests,%{ldap_base}
ldap_default_dn: uid=default-dialup,%{ldap_base}
ldap_regular_profile_attr: dialupregularprofile
#
# If set to yes then the HTTP credentials (http authentication)
# will be used to bind to the ldap server instead of ldap_binddn
# and ldap_bindpw. That way multiple admins with different rights
# on the ldap database can connect through one dialup_admin interface.
# The ldap_binddn and ldap_bindpw are still needed to find the DN
# to bind with (http authentication will only provide us with a
# username). As a result the ldap_binddn should be able to do a search
# with a filter of (uid=<username>). Normally, the anonymous (empty DN)
# user can do that.
#ldap_use_http_credentials: yes
#
# If we are using http credentials we can map a specific username to the
```

```
# directory manager (which usually does not correspond to a specific username)
#
#ldap_directory_manager: cn=Directory Manager
#ldap_map_to_directory_manager: admin
#
# Uncomment to enable ldap debug
#
#ldap_debug: true


#
# can be one of mysql,pg where:
# mysq: MySQL database (port 3306)
# pg: PostgreSQL database (port 5432)
#
sql_type: mysql
sql_server: localhost
sql_port: 3306
sql_username: dialup_admin
sql_password: XXXXXX
sql_database: radius
sql_accounting_table: radacct
sql_badusers_table: badusers
sql_check_table: radcheck
sql_reply_table: radreply
sql_user_info_table: userinfo
sql_groupcheck_table: radgroupcheck
sql_groupreply_table: radgroupreply
sql_usergroup_table: usergroup
#
# Uncomment to enable sql debug
#
#sql_debug: true
#
# If set to yes then the HTTP credentials (http authentication)
# will be used to connect to the sql server instead of sql_username
# and sql_password. That way multiple admins with different rights
# on the sql database can connect through one dialup_admin interface.
#sql_use_http_credentials: yes


#
# true or false
#
sql_use_user_info_table: true
sql_use_operators: true
#
# Set this to the value of the default_user_profile in your
# sql.conf if that one is set. If it is not set leave blank
# or commented out
#sql_default_user_profile: DEFAULT
```

```
#
#
sql_password_attribute: User-Password
sql_date_format: Y-m-d
sql_full_date_format: Y-m-d H:i:s
#
# Used in the accounting report generator so that we
# don't return too many results
#
sql_row_limit: 40
#
# These options are used by the log_badlogins script
#
# Set the sql connect timeout (secs)
sql_connect_timeout: 3
# Give a space separated list of extra mysql servers to connect to when
# logging bad logins
#sql_extra_servers: sql2.company.com sql3.company.com

counter_default_daily: 14400
counter_default_weekly: 72000
counter_default_monthly: none
```

APPENDIX I

dialup_admin is a web based administration interface for the freeradius radius server.
It is written in PHP4 (although the files have an extension of php3 for historical reasons).
It is modular and right now it assumes that user information is stored in an ldap server
or an sql database and accounting in an sql server.

There are also a few more things included:
* sql/badusers.sql: It will create a table named badusers which can be used to hold the history for badusers (date,action)
* sql/userinfo.sql: It will create a table named userinfo which contains personal information for users stored in the sql database. This can be used if a sql database is used to store user profiles (not ldap).
* bin/log_badlogins: It will constantly check the radiusd.log file and add all login-incorrect, invalid-user, outside of allowed login time and multiple-login cases to the radacct table with acctstarttime=acctstoptime, acctsessiontime=0 and acctterminatecause containing the reason. If the failing module sent back a module message then it will also appear in the terminate cause. You will need to setup the $mysql and $tmpfile variables to suite your mysql installation and needs. Perl module Date::Manip is needed by the script.
* bin/clean_radacct: It will delete all entries in the radacct table with a starttime > 1 day and stoptime = 0. It will not do an harm even if it deletes valid entries since radiusd will fall back to insert if update fails.

The structure of the tree is:
conf:: Contains various configuration files.
conf::admin.conf=>
The main configuration file. The directives used should be easily understood
conf::config.php3=>
Just a helper php4 for reading the admin.conf file.
conf::default.vals=>
Contains the default values for check and reply items. If you also use the users file except for the ldap/sql databases fill in the default values you have inserted for the users.Else comment them out.
conf::sql.attrs=>
Contains inmformation about the sql attributes. This is used by the accounting report generator (accounting.php3).
conf::user_edit.attrs=>
Contains the attributes that user_edit.php3 will show. Just uncomment those which you want to be displayed.
conf::extra.ldap-attrmap=>
Contains a few items not included in ldap.attrmap which are used only by dialup_admin Things like User-Password and Dialup-Lock-Msg. Use none for attributes that are not known by the radius or ldap server.
conf::sql.attrmap=>
Contains the Attribute map for the sql database. Use none for attributes that are not known by
the radius server.
conf::auth.request=>
Contains value pairs sent in the test packets which are sent from the test user/server pages.

htdocs:: Contains the basic php files
htdocs::index.html=>
The main index file. Just contains the frames tags
htdocs::content.html=>
Change this file to include the greeting of your choise
htdocs::buttons.php3=>
This will open the corresponding buttons html file in the html/buttons folder
htdocs::style.css=>
CSS style file. Change it to match your preferences
htdocs:: user_state.php3 =>
It can be used by outside pages to get a quick
overview of the status of a user. It will return the following fields
separated by new lines:
account_status(active or inactive),lock message,weekly limit,daily limit,
weekly used,weekly connections,daily used,daily connections
htdocs:: user_finger.php3 =>
It will finger the nas(es) and show the logged in users. If an argument server is passed then
it will only show users for the specific access server.

html:: Contains the html code for a few pages
html::user_admin.html.php3 =>
html code for the user_admin page
html::stats.html.php3 =>
html code for the stats page
html::user_toolbar.html.php3 =>
the user toolbar (show,edit,accounting...) which appears in almost all pages
html::group_toolbar.html.php3 =>
the toolbar shown in the group manipulation pages
html::/buttons
contains a folder default which contains a buttons.html.php3 file. When a user is connecting
with http authentication we first try to open the file in the folder <username>. If that fails
we open the file in the default folder. That way each admin can have a different view of the
buttons
bar (for example see different finger pages).

lib:: Contains the library items
lib::ldap=>
The ldap files
lib::lang=>
Allow for multiple languages to be supported
lib::crypt=>
Allow for multiple encryption schemes (for user passwords)
lib::sql=>
Contains the sql library files. There is also a directory drivers
which contains functions for various sql databases (mysql and
postgresql for the time being).

The rest: The rest are the php4 files. Normally, you should not need to change anything in them.