

Deakin University RADIATOR deployment

Introduction

This page will describe in some detail how RADIATOR is set up at Deakin, for all RADIUS based authentication, including Eduroam and our own WPA[2] network. It is fairly verbose, and not just about Eduroam, so you can see what is involved in using it for your RADIUS service. If you want to know more about how we configure it for Eduroam, please see the sections below about WPA[2]. Note, that a lot of this has been ripped from our internal doco, to save me re-writing it. If you see anything you think may be a security issue, or want stuff clarified, drop me a line john.stevens@deakin.edu.au.

Radiator is a perl based RADIUS server that is highly extensible. IT is produced by Open Systems (<http://www.open.com.au/radiator/index.html>).

If you are going to e-mail support, you MUST include a Trace 4 log snippet of the problem, otherwise you will wait a few hours for them to ask you for it. They really need this to be of help. To effect a Trace 4 log, you must alter the config file and change the Trace line near the top of the file to read Trace 4. Restart Radiator to put the change in effect. Run whatever you need to produce output, and save the relevant sections of the log file to another file to send off. Don't forget to put the Trace level back when you are done.

How it works @ Deakin

Basically, Clients connect to a RADIUS server for authentication. In order to be allowed to connect, Radiator must know about the client. See Configuration Notes for details on how to do this. Radiator itself will pass the supplied credentials or accounting information to either our SQL based User Management System (most common), or our Active Directory (wireless/eduroam only) for verification.

In the case of our SQL based UMS authentication, all Radiator instances are configured to try our production Database first, and if a response is not immediately received, it will try a backup copy running on another server on another campus. This allows for an extra layer of redundancy, and allows ANY RADIUS server to be used for any authentication in the event they are required. It also means that a local RADIUS (comms) server can be the first RADIUS server tried by clients, with a fallback of the most immediately adjacent RADIUS server.

In the event of a failure to connect to the Production UMS, the Backup UMSB is tried. There is a fall back delay of 10 seconds before Production UMS will be tried again. Basically this ensures that the Production UMS is used first for all access and accounting request handling, and worst case scenario is that the Backup UMS will be used until 10 seconds after Production is contactable again.

Note: If both Production and Backup are unreachable, accounting and access dependant on UMS will fail.

Wireless (802.1x or deakinsecure) authenticates against the first available AD domain controller. If UMSP/B are unavailable, authentication will still work, as this will use the local domain controller, or one of the three other domain controllers in our network. However, as all accounting is into UMS, accounting requests will fail.

Configuration Notes

Configuration files are located in /etc/radiator/radius.cfg on RHEL. All global parameters should be listed in the start of the configuration file.

Basic Guidelines

1. Add comments to describe what things are or what they do. Comments are anything after a # character.
2. Use IdenticalClients statements in a single Client clause, rather than creating LOTS of Client clauses.
3. Use Identifier statements in Client clauses rather than DefaultRealm statements.
4. Use Handler clauses to process requests rather than Realm clauses. `<Handler Client-Identifier=clientIdentifier>` will match a Client clause with Identifier clientIdentifier statement. It is more flexible than using Realms.
5. Make sure Handlers called after other Handlers appear before them in the config. See below for details.
6. DO NOT HAVE A DEFAULT HANDLER!!! You should NEVER handle any RADIUS message you are not explicitly handling.

Clients

All clients that are to connect to a RADIUS server must be listed in the configuration. Client clauses consist of a fqdn/IP adress and a shared secret as a minimum. Client clauses also have the capability to use IdenticalClients statements so that you can configure multiple similar clients with one Client clause. Note: You can store clients in a SQL database, but we prefer to allow radiator to be non-reliant on external Network Services where possible, for reliability sake.

For example:

```
# Modems and other general RADIUS clients
<Client somemodembank1.deakin.edu.au>
  IdenticalClients someconsoleserver1.deakin.edu.au
  IdenticalClients somehost1.deakin.edu.au
  IdenticalClients somemodembank2.deakin.edu.au
  IdenticalClients somehost2.deakin.edu.au
  IdenticalClients somemodembank2.deakin.edu.au
# Socc client hosts
  IdenticalClients socdev.deakin.edu.au
  IdenticalClients socmain.deakin.edu.au
  IdenticalClients socsat1.its.deakin.edu.au
  IdenticalClients socsat2.its.deakin.edu.au
  IdenticalClients socsat3.its.deakin.edu.au
  IdenticalClients socsat4.its.deakin.edu.au
  Secret SomethingReallySecret
</Client>
```

Clients can also be assigned a default Realm in case a username is passed without a FQDN appended (like `user@deakin.edu.au`). Although we still do this in our Radiator configs, it is NOT considered Best Practice. It is better practice to use Client-Identifier statements to uniquely identify Requests from a Client clause. The Client-Identifier is something that can be used to match a Handler clause for greater control on how requests are handled. This is used for our wireless access points.

Handlers

Handler clauses actually define which request is handled based on various criteria. A Request is processed through the handlers in the order that they are defined, until a match is found. If a match is not found, it is handled by the default handler if one is defined. A default handler is a Handler clause with No criteria to match, and MUST be the last Handler defined.

When a Handler is found that matches a request, the Handler will use ~AuthBy clauses to decide if the request should be Accepted, Rejected, or Ignored.

The most important thing to note about Handler clauses is they are assessed in the order they are configured in the file. If a Handler clause is referenced by another Handler, it should appear before it in the config. Why? Well, think about it. If a Handler passes the Request to another handler, this is actually a new Request and must be processed through all the Handlers again. If you have the one that matched first before the one you want to pass it to, the original Handler will match, and you are in an infinite loop. So, be careful about this.

Handler clauses should be used in preference to Realm clauses. Although they are similar, Handlers are far more flexible. Our current configs include Realm clauses to match DefaultRealm statements in Client clauses, but this is contrary to Best Practice and will be phased out (and hence Realm clauses will not be discussed in this page).

Handlers consist of at least one AuthBy clause as a minimum.

AuthBy

There are various methods of AuthBy available to Radiator, and some depend on the host OS, or other perl modules/libraries to function. At Deakin, we use AuthBy SQL to authenticate users against UMS, and AuthBy NTLM to authenticate users and hosts against AD as part of the DeakinSecure 802.1x wireless network (and eduroam). An example of an AuthBy SQL is below:

```
<Handler Realm=VPN>
  RewriteUsername      s/([@]+).*/$1/
  <AuthBy GROUP>
    AuthByPolicy        ContinueUntilAccept
    <AuthBy SQL>
      <Snipped. See the manual for info on how to configure this for your SQL db>
    </AuthBy>
    <AuthBy SQL>
      <Snipped. See the manual for info on how to configure this for your SQL db>
    </AuthBy>
  </AuthBy>
</Handler>
```

This shows just how complex an AuthBy clause can get. Note that there is an AuthBy GROUP around two AuthBy SQL clauses. This allows Radiator to automatically fail over to Our Backup UMS SQL server in the event of the Production server being unavailable, and automatically fall back 10 seconds (worst case) after Production becomes available again.

RewriteUsername

The RewriteUsername statment can appear in Client, Handler, and Realm clauses. It is essentially a perl substitution definition (regexp) and allows things like turning all characters to lowercase, or removing stuff before or after a username. If used in a Client clause, the username will be altered at that point and passed further through the Handlers or Realms. If it is defined in a Handler or Realm, it will only be in effect during that Handler or Realm. An example of usage would be:


```
#Handler for the 802.1X inner authentication for Deakin users and machines if using PEAP.
<Handler TunnelledByPEAP=1,User-Name=/(^AD\\|@deakin\.edu\.au|^host\/.*\.ad\.deakin\.edu\.au)/>
  #Get rid of the AD\bit.
  RewriteUsername s/^ad //
  #Auth against AD with ntlm_auth
  <AuthBy NTLM>
    EAPType MSCHAP-V2
    Domain AD
    UsernameMatchesWithoutRealm
  </AuthBy>
  #Strip any prior cisco-avpair attributes and add the DeakinSecure SSID.
  StripFromReply cisco-avpair
  AddToReply cisco-avpair="ssid=DeakinSecure"
</Handler>
```

This rewrites the username to remove any leading AD\ . We are specifying the domain to be AD in the ~AuthBy NTLM clause.

Accounting

Radiator, like all RADIUS servers can be used to log accounting information about Authorisation requests. Here at Deakin, we currently use Accounting only in AuthBy SQL clauses that use a SQL table to store accounting information in. Accounting is, obviously, handled by AuthBy clauses, and each clause can have it's own accounting method. By default, an AuthBy clause will not log accounting information. For details on what methods of accounting you have available, see the documentation.

AD and Radius

All our RADIUS servers utilise ntlm_auth (part of SAMBA) to authenticate access requests form wireless clients. In order to accomplish this, the comms servers need to be "joined" to the Active Directory domain AD, and need to be able to communicate with their campus Domain Controller.

Wireless WPA[2] and Eduroam

Outer Handler

This Handler clause handle requests coming from the Identifier wirelss-8021x Client clause, or all configured Access Points. It handles all requests using an AuthBy File clause. The file that it auths against is the /etc/radiator/users file which contains the user anonymous which accepts any password and is used for WPA/2 outer Authentication requests.# Start DEAKIN OUTER HANDLER

```
#Handler for the Outer authentication of Deakin user/host names. Note: wdsclients is for the access points.
<Handler Client-Identifier=wireless-8021x,User-Name=/(^AD\\|@deakin\.edu\.au|^host\/.*\.ad\.deakin\.edu\.au)/>
  #We use this file with an anonymous user that we just say OK to.
  <AuthBy FILE>
    Filename %D/users
    #We allow all of these even though we really want PEAP. TTLS and LEAP are used by Macs and WDS
    EAPType PEAP,TTLS,TLS,LEAP
    #Use whatever username was passed to us as the EAP Anonymous user.
    EAPAnonymous %0
    #
    #Deakin Certificates
    #
    EAPTLS_CAPath /etc/radiator/certificates
    EAPTLS_CertificateFile /etc/radiator/certificates/deakinsecure.crt
    EAPTLS_CertificateType PEM
    EAPTLS_PrivateKeyFile /etc/radiator/certificates/deakinsecure_key.pem
    EAPTLS_MaxFragmentSize 1000
    AutoMPPEKeys
    #We use PEAPVersion 0 because the Macintosh clients don't like version 1
    EAPTLS_PEAPVersion 0
    #This is added for Apple Macintosh Airport Extreme adapters
    EAPTTLS_NoAckRequired
  </AuthBy>
</Handler>
#End DEAKIN OUTER HANDLER
```

Note that this handler utilises the [deakinsecure.deakin.edu.au](#) certificate, which is externally validated and signed by Thawte. This allows clients to utilise this as the shared key to encrypt the outer traffic. You will need to modify this to your own certificate's location. This handler will add an attribute to the request of either TunnelledByPEAP or TunnelledByTTLS, depending on which EAP type the client uses for outer authentication.

Inner Handlers

We utilise two inner handlers for the authorisation of Deakin users and hosts. Why? Because we accept EAP-PEAP and EAP-TTLS as the outer authentication layer, we need a handler for each of these. The Deakin handlers are like this.# Handler for the 802.1X inner authentication for Deakin users and machines if using PEAP.

```

<Handler TunnelledByPEAP=1,User-Name=/(^AD\\|@deakin\.edu\.au|^host\/.*\.ad\.deakin\.edu\.au)/>
#Get rid of the AD\bit.
RewriteUsername s/^ad//
#Auth against AD with ntlm_auth
<AuthBy NTLM>
    EAPType MSCHAP-V2
    Domain AD
    NtlmAuthProg /usr/bin/ntlm_auth -s /etc/samba/deakin-winbindd.conf --helper-protocol=ntlm-server-1
    UsernameMatchesWithoutRealm
</AuthBy>
#Strip any prior cisco-avpair attributes and add the DeakinSecure SSID.
StripFromReply cisco-avpair
AddToReply cisco-avpair="ssid=DeakinSecure"
</Handler>
#Handler for the 802.1X inner authentication for Deakin users and machines, if using TTLS.
<Handler TunnelledByTTLS=1,User-Name=/(^AD\\|@deakin\.edu\.au|^host\/.*\.ad\.deakin\.edu\.au)/>
#Get rid of the AD\bit.
RewriteUsername s/^ad//
#Auth against AD with ntlm_auth
<AuthBy NTLM>
    EAPType MSCHAP-V2
    Domain AD
    #NtlmAuthProg /usr/bin/ntlm_auth -s /etc/samba/deakin-winbindd.conf --helper-protocol=ntlm-server-1
    UsernameMatchesWithoutRealm
</AuthBy>
#Strip any prior cisco-avpair attributes and add the DeakinSecure SSID.
StripFromReply cisco-avpair
AddToReply cisco-avpair="ssid=DeakinSecure"
</Handler>

```

These handlers use AuthBy NTLM clause to authenticate a user/host. The important parts to note are the Handler definition and the StripFromReply and AddToReply statements. The Handler definitions specify the requests that each handler will match and process. The User-Name attribute of the RADIUS request MUST match usernames of the form AD\username, [username@deakin.edu.au](#), or [host.machinename.ad.deakin.edu.au](#), which is a host authentication request. Then each handler will also match the type of Tunnel the request is being passed in, either an EAP-PEAP or an EAP-TTLS tunnel. This is by looking for the TunnelledByTTLS=1 or TunnelledByPEAP=1 attributes that the outer handler added to the request. If the AuthBy clause succeeds, ie the user is authenticated, then the handler that authenticates it will send the allowed SSID back to the access point. This is accomplished by adding the cisco-avpair attribute to the Reply with the AddToReply statement. While it is really not necessary to have the StripFromReply statement, it is considered best practice in Radiator to make sure attributes don't exist before you add any.

Note also that these handlers use the NtlmAuthProg statement to tell ntlm_auth to use the /etc/samba/deakinwinbindd.conf configuration file. This allows ntlm_auth to continue to function even when samba is started on a comms host, most likely so that it can take over from a failed file server. The samba config for domain authentication used by ntlm_auth, and our standard samba config are mutually exclusive.

Eduroam Guest Handler

Eduroam requires that a client authenticate with their HOME institution through the Eduroam RADIUS proxy servers, and this means that we must proxy any non-Deakin user Access-Requests to the National Eduroam Servers. The proxy will handle all tunnel setup and access control, except for the final Access-Accept message. If a user successfully authenticates through the Eduroam servers, we add the cisco-avpair ssid=eduroam to make sure they can only connect to the Eduroam SSID. See the handler below for details. This Handler MUST appear after the Deakin outer handlers in the Radiator config file (/etc/radiator/radius.cfg, as these handlers are processed in order. Because we are assessing the User-Name RADIUS attribute, and Handler definitions are assessed in the order they appear in the config file, the more defined Deakin Handler test MUST appear before less defined Eduroam Handler test. # Start EDUROAM CLIENT HANDLER

```

#Handler for Eduroam usernames of the form user@domain, that come from one of the wireless-8021x clients (Access Points).
#We proxy these straight out to Eduroam's National RADIUS servers, and they take care of it from there.
<Handler Client-Identifier=wireless-8021x,User-Name=/@/>
#Eduroam
<AuthBy RADIUS>
    Host 202.158.207.10,202.158.207.11
    AuthPort 1812
    AcctPort 1813
    Secret somethingreallysecret
    Retries 1
</AuthBy>
#Strip any prior cisco-avpair attributes and add the eduroam SSID.
StripFromReply cisco-avpair
AddToReply cisco-avpair="ssid=eduroam"
</Handler>
End EDUROAM CLIENT HANDLER

```

Eduroam Outer Handler

Because we also handle requests from Eduroam servers (we have Client clauses for them as well), we also need an outer handler for requests they are sending to us to authenticate Deakin users on other organisations campuses. The outer handler looks very much like our normal outer handler, except for one Major difference.

```

<Handler Client-Identifier=Eduroam-Server>
<AuthBy FILE>
    Filename %D/users

```

```

#We allow all of these even though we really want PEAP.  TTLS and LEAP are used by Macs and WDS
EAPType PEAP,TTLS,TLS,LEAP
Use whatever username was passed to us as the EAP Anonymous user.
EAPAnonymous %0
#
# Deakin Certificates
#
EAPTLS_CAPath /etc/radiator/certificates
EAPTLS_CertificateFile /etc/radiator/certificates/deakinsecure.crt
EAPTLS_CertificateType PEM
EAPTLS_PrivateKeyFile /etc/radiator/certificates/deakinsecure_key.pem
EAPTLS_MaxFragmentSize 1000
AutoMPPEKeys
#We use PEAPVersion 0 because the Macintosh clients don't like version 1
EAPTLS_PEAPVersion 0
#This is added for Apple Macintosh Airport Extreme adapters
EAPTTLS_NoAckRequired
</AuthBy>
#Strip any VLAN stuff from the reply.
StripFromReply Tunnel-Type,Tunnel-Medium-Type,Tunnel-Private-Group-ID,cisco-avpair
</Handler>

```

So how does that work? Well, even though the actual authentication is performed by the inner handler, the RADIUS Reply MUST pass back through all handlers before being returned to the Initiating device (NAS). So the Access-Accept message generated by the inner handler, with it's added VLAN assignment, comes back to this handler where the VLAN assignment is stripped. The Access-Accept is then passed back to the NAS.

Notice the StripFromReply of the VLAN information? We use our standard inner handlers to actually do the authentication. But these add VLAN assignments. What happens if we pass this to another institution's AP? Well, we don't know and we don't want to find out, so we strip it out.