How to Configure Govroam RRPS to proxy sites

This is not a line by line configuration guide for specific RADIUS software. The number of permutations and combinations of wireless systems, RADIUS servers and client makes that close to impossible in a single document.

What this documents intends to cover is the general case for configuring an RRPS to proxy traffic from members' sites.

The Regional RADIUS Proxy Server (RRPS) has two main jobs:

1. Proxy authentication requests for known realms between Organisational RADIUS PROXY Servers (ORPS).
2. Proxy authentication requests for unknown realms to the National RADIUS Proxy Servers (NRPS).

There are a number of additional requirements such as:

- Logging of transactions for audit purposes.
- Sending logs to Jisc for statistical purposes.
- Filtering out inappropriate requests.

To achieve these there are a number of steps that need completing.

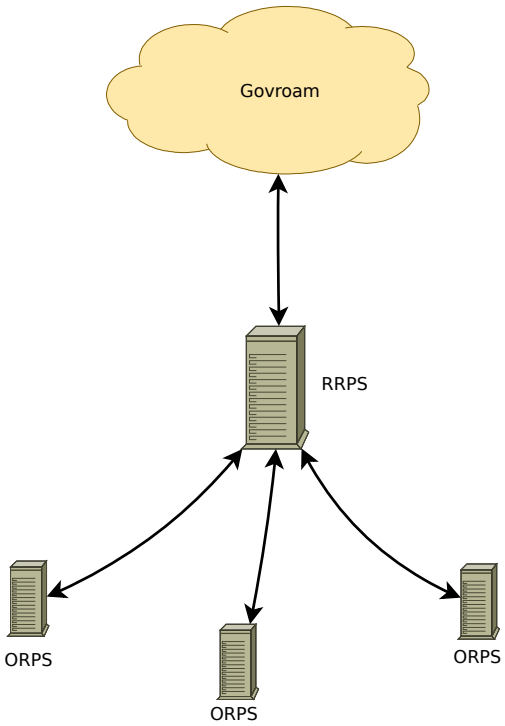

Illustration 1: Basic RADIUS Layout

Inter-server communication

The most fundamental configuration is to allow two RADIUS servers to talk to each other, whether they are the ORPS and RRPS or the NRPS and RRPS. The approach is identical and for each RADIUS server you will need:

- IP address/hostname
- Shared Secret

There is normally a Shared Secret for each pair of RADIUS servers that need to communicate. For Illustration 1 that would mean four separate shared secrets (assuming 1 NRPS in the cloud). If there were three RRPS and four NRPS then the number would rise to 21 (twelve between RRPS and NRPS and three each for the ORPS). When planning it's worth creating a table to ennumerate the configuration combinations:

Server A	Server B	Shared Secret
RRPS 1 (10.10.10.1)	NRPS 1 (2.2.2.1)	randomstring1
RRPS 1 (10.10.10.1)	NRPS 2 (2.2.2.2)	randomstring2
RRPS 1 (10.10.10.1)	NRPS 3 (2.2.2.3)	randomstring3
RRPS 1 (10.10.10.1)	NRPS 4 (2.2.2.4)	randomstring4
RRPS 2 (10.10.10.2)	NRPS 1 (2.2.2.1)	randomstring5
…		
RRPS 3 (10.10.10.3)	ORPS 3 (10.20.30.1)	randomstring21

It will become confusing quickly if you don't document as you along. A sensible naming/numbering scheme would also help. Whether you're using a GUI or editing text files understand the structure and try to use sensible string and descriptions to keep it clear. Over time the configuration could grow considerably.

RADIUS servers normally differentiate between Clients and Servers. In RADIUS terminology a Client is anything that passes a RADIUS query to a RADIUS server, including other RADIUS servers. So in illustration 1, the three ORPS will all Clients to the RRPS. However, the RRPS will also be a Client to the ORPS. The NRPS will be Clients to the RRPS, and the RRPS Client to the NRPS.

The RRPS also has to know where to proxy requests to and the RADIUS servers which receive them are known as Remote Servers, Home Servers or similar names. As with the Client, the Server status is relative.

So, ensure that you have all the associated RADIUS servers configured as both Client and Server. There are cases when you might not want them to be both[1].

Tip: if you're using text files then look for 'include' options and consider splitting the configuration into one file per site.

Testing can be tricky as the RADIUS servers need to generate a request of some sort and be able to see the response.

Tip: Don't forget your firewall rules – port 1812/UDP needs to allow incoming connection for each RRPS from each NRPS or ORPS. The above example table would help structure this.

If you can create an account locally on the RADIUS server and add a local authentication option to the configuration then the connected sites might be able to use (non-EAP) authentication tests (basic PAP or MSCHAP) to prove that the various RADIUS servers can communicate, thus

1 If a RADIUS server is proxy from a Visited Only site then it should only be a Client. If the Remote Server is just doing authentication (i.e. it's an IdP) then it only needs to be a Server.

proving that the Shared Secrets are right and the firewall rules are in place. This is normally a good initial test before sites move on to more complex EAP testing. Nothing is more frustrating than spending time trying to fix a perceived problem with something complex only to find that the problem is in something simple.

Realm Proxying

As stated above proxying realms is the main purpose of the RRPS. So once the interserver communication is working then this is the next step.

It doesn't matter which you do first, known or unknown realms. You might find it easier to test with other sites in your Federation, or you might prefer to work with Jisc to get the communication with the NRPS working.

> **Tip: Whilst the RRPS is proxing EAP, it doesn't need to process EAP protocols. It just passes on RADIUS packets that contain EAP information somewhere.**

The RADIUS packet being proxied will contain a number of different attributes. One of which is User-Name. Proxies will only see the outer identity. The inner identity (provided by the end user's client) is encrypted within the RADIUS data and will never be visible to the proxy. The purpose of the outer identity is to provide the proxies with a realm to be used to route the request to the right place. The User-Name attribute will be structured as 'string@realm' e.g. 'fred@camford.gov.uk'. The user part 'fred' is irrelevant to the proxying. The realm part 'camford.gov.uk' is the important part as provides information to the RRPS as to the final destination of the request.

The configuration of the RRPS needs the following basic logic:

IF Realm eq 'Arealm' THEN ProxyTo RemoteServer A

ELSE IF Realm eq 'Brealm' THEN ProxyTo RemoteServer B

ELSE IF Realm eq 'Crealm' THEN ProxyTo RemoteServer C

...

ELSE ProxyTo NRPS

How this is achieved is very software specific and often there are multiple ways of achieving the same thing.

Where there are multiple RemoteServers handling a particular realm then there is usually a mechanism within the RADIUS server where the Remote Servers are grouped together and the logic would be 'ProxyTo RemoteServerGroup A' instead. Often there is are a number of load balancing options within the pool (round robin, least used, source IP hash etc.).

Tip: Some load balancing works better than others. An EAP conversation is normally 12 interactions so MUST be processed by the same pair of servers, splitting the request across multiple will just fail.

The default destination should be the NRPS configuration i.e. if the realm fails to match any of the configured realms then send the request upwards to the NRPS. This ought to represent visitors to your site who need their credentials authenticating elsewhere.

Now you should have a basic but functional proxying system. Testing EAP can be tricky as there aren't many tools out there. Normally the best way is to build up a set of systems and test each as you go. e.g. start with a wireless system using a captive portal and local accounts. Add an IdP with local credentials and use PAP or MSCHAP. Enable 802.1x on the wireless system. Use EAP-PEAP on the IdP. Add an external store of credentials (e.g. AD). Insert a proxy between wireless system and IdP.

Additional Requirements

If all the above works then you should be comfortable enough to move onto the additional requirements.

Local Logging

It is a requirements of Govroam to keep transaction logs for a period of three months. This forms part of the audit trail for tracing users who may have misbehaved. The Tech Spec specifies exactly what must be kept. How you decide to format your logs is entirely up to you. Some people keep them by day, some by source, some per server, but as long as those logs exist and can be used to provide a trail then that's all that's important.

Tip: RADIUS logs can be quite large so consider compressing them. Ensure that you've accounted for 90 days worth.

Some sites also generate summaries as the default log formats can be difficult to work with. F-TICKS is a format that was developed specifically to handle eduroam logs. It's a single line per authentication and lends itself well to both syslog and parsing by machines.

Remote Logging

Whilst it's not (currently) as requirement Jisc would very much appreciate information about roams being sent to them. By default Jisc can only see the roams when someone moves to a different Federation entirely so miss all the data associated with roams between sites within a Federation. Thus Jisc are asking that RFOs send F-TICKS logs of such roams to them. This can be done with syslog if the RADIUS server supports it. Filebeat/Winfilebeat is a tool that can read

logs and pass the data (suitably) filtered to the Log Server at Jisc.

Jisc doesn't need, or want, any personal identifiable information. The logs only need to contain the basic information about originating site and realm along with a date stamp. Again F-TICKS fulfills these criteria well.

Realm Filtering

The section of proxying makes assumptions about the quality of the data being used as realms. In an ideal world only valid realms would be seen in the outer User-Name attribute. However, the end-user devices are often configured by end-users who don't necessarily know what should be going on and some devices seem to pick up default values. Thus the realm is often populated with completely invalid realms. In fact the ratio of invalid to valid is about 1000:1.

By far the most common invalid realm is NULL. This means that the User-Name is formated 'string' rather than 'string@realm'. Less common is 'string@'. These are relatively easy to filter out. IF Realm is NULL Discard, IF Realm eq '' Discard etc.

Then there are syntactically invalid realms 'camford..gov.uk', '@@camford.gov.uk', '@camford.gov.', '@camford'. These can be filtered out using pattern matches.

After that it gets harder. It's very common to see '.3gppnetwork.org' in a realm, which doesn't exist in Govroam. Spelling mistakes account for many '@cmford.guv.uc' with many, many variations.

Then there are valid realms but not for Govroam e.g. 'manchester.ac.uk' where a student is trying to use Govroam rather than eduroam.

Then there are realms which are questionable e.g. '@nhs.uk'. Does this exist or not?

It is expected that administrators check their logs periodically, spot any invalid realms being sent to the NRPS and add them to a blacklist of some sort.

Visited Only

In some cases sites have no eligible users of their own but wish to offer Govroam services for visitors (e.g. Universities, care homes). The configuration of RADIUS servers is almost identical to the above. As explained above one of the functions of the RRPS is to proxy requests from local users to the appropriate ORPS, however this no longer applies for Visited-Only sites. There are no local users and it is assumed that everyone using Govroam is a visitor who needs authentication via an off-site IdP. Thus all (except those covered by Realm Filtering) requests are proxied to the NRPS.

Essentially the implementation is as above but omitting any configuration for sending requests to other ORPS or IdPs i.e. the ORPS clients are just the wireless controllers and the NRPS clients are just the ORPS.