Install an RP Proxy on Debian 8

On this page you will find instructions on how to set up a Moonshot RP Proxy on Debian 8 (Jessie) using FreeRADIUS. It also installs and configures the Trust Router client, if you are going to use the Trust Router infrastructure.

Contents

If your organisation already has a Moonshot Identity Provider, this can also be used as a Moonshot RP Proxy - you may not need to install a Moonshot RP Proxy as well.

Complexity of Installation

Many of the steps outlined below are currently necessary, but we realise the install should be simpler. As the software matures and the packaging improves, we will to make this process easier with fewer manual steps required.

1. System Preparation

1.1. Install Debian 8

The first thing that is required is a Debian 8 machine - this can be physical or virtual.

1. Install Debian 8 (Jessie) via usual mechanism (e.g., netboot CD, ISO in VMware/VirtualBox or the DVD image.
2. Choose the following server install options: "Debian desktop, SSH server, Standard system utilities".
3. Create/choose a secure root password and an initial system user account.
4. Once installed, make sure you run an `apt-get update` and `apt-get upgrade` to ensure your system is fully up to date.

Tip

We would recommend using LVM when disk partitioning to allow easier partition/disk expansion on a live system.

Warning

After install, you will want to secure/lockdown the server as best practice dictates - for both the server and any extra software installed. This is beyond the remit of this guide but there are many guides available that provide information on securing your Debian servers and applications.

1.2. Configure Debian 8

Next, there are a few Debian configuration options that need to be set in advance.

1.2.1. Networking configuration

For production deployments, it is recommended that the machine be assigned a static IP address.

For Debian networking information please refer to the Debian documentation: https://wiki.debian.org/NetworkConfiguration

1.2.2. Firewall configuration

The following ports are required to be accessible from the outside world, both in the local firewall and in any external firewalls:

- 2083/tcp (for RadSec connections to other Moonshot entities)
- 12309/tcp (for Trust Router client connections - if using the Trust Router to broker trust relationships between entities)

Here are sample firewall rules that establish incoming and outgoing rules to both the Test and Live (Jisc Assent) Moonshot trust router infrastructures. If you connect to another Trust Router, adjust these rules to suit:

IP Tables sample firewall rules (Jisc Assent)

```
-A INPUT -m state --state NEW,ESTABLISHED,RELATED -m tcp -p tcp -s 0/0 --dst <IdP/RP Proxy IP address> --
dport 2083 -j ACCEPT
-A OUTPUT -m state --state NEW,ESTABLISHED,RELATED -m tcp -p tcp -s <IdP/RP Proxy IP address> --dst 0/0 --
dport 2083 -j ACCEPT
 -A INPUT -m state --state NEW,ESTABLISHED,RELATED -m tcp -p tcp -s
212.219.179.130,212.219.179.131,212.219.179.138,212.219.179.146 --dst <IdP/RP Proxy IP address> --dport
12309 -j ACCEPT
-A OUTPUT -m state --state NEW,ESTABLISHED,RELATED -m tcp -p tcp -s <IdP/RP Proxy IP address> --dst
212.219.179.130,212.219.179.131,212.219.179.138,212.219.179.146 --dport 12309 -j ACCEPT
```

IP Tables sample firewall rules (Test Network)

```
-A INPUT -m state --state NEW,ESTABLISHED,RELATED -m tcp -p tcp -s 0/0 --dst <IdP/RP Proxy IP address> --
dport 2083 -j ACCEPT
-A OUTPUT -m state --state NEW,ESTABLISHED,RELATED -m tcp -p tcp -s <IdP/RP Proxy IP address> --dst 0/0 --
dport 2083 -j ACCEPT
-A INPUT -m state --state NEW,ESTABLISHED,RELATED -m tcp -p tcp -s 13.79.134.211,13.79.128.103,52.169.31.1
04 --dst <IdP/RP Proxy IP address> --dport 12309 -j ACCEPT
-A OUTPUT -m state --state NEW,ESTABLISHED,RELATED -m tcp -p tcp -s <IdP/RP Proxy IP address> --dst 13.79.
134.211,13.79.128.103,52.169.31.104 --dport 12309 -j ACCEPT
```

1.3. Add the Moonshot Repository

1. Add the Moonshot Debian Jessie repository to your system. To do this, run the following command (as root, or using sudo):

   ```
   $ echo "deb http://repository.project-moonshot.org/debian-moonshot jessie main" > /etc/apt/sources.list.d
   /moonshot.list
   ```

2. Install the Moonshot GPG key:

   ```
   $ wget -O - http://repository.project-moonshot.org/key.gpg | apt-key add -
   ```

   Verifying the Moonshot GPG key

   If you wish to verify the Moonshot GPG key's validity and integrity, please see the Packaging GPG Key for further details.

3. Update the apt cache with the new repository information:

   ```
   $ apt-get update
   ```

2. Install the Moonshot RP Proxy

We're now ready to install the Moonshot software and its required dependencies. Install the software by running the following command:

```
$ apt-get install freeradius-abfab freeradius-utils moonshot-gss-eap moonshot-ui moonshot-trust-router dbus-x11
```

If you try to start FreeRADIUS at this point, it will not currently start successfully as the certificates it requires have not been generated - they are created in step 4.1 below.

3. Configure the Moonshot RP Proxy

Next, we need to configure the Moonshot RP.

3.1. Configure FreeRADIUS

3.1.1. Certificates

We need to get FreeRADIUS to create some private and public keys to use for its RadSec connections. Create and install the certificates by doing the following (as root).

1. Change into the `/etc/freeradius/certs` directory

```
$ cd /etc/freeradius/certs
```

2. Edit the certificate generation properties in *client.cnf*, *server.cnf*, and *ca.cnf* as follows:

   a. In the `ca.cnf` file:
      i. In the `[ req ]` section, add `encrypt_key = no`
      ii. In the `[CA_default]` section, change the default_days from 60 to a higher number (this is how long the certificates you create will be valid for). When the certificates expire, you will have to recreate them.
      iii. in the `[ certificate_authority ]` section, change all of the parameters to match those of your organisation. e.g.

```
[certificate_authority]
countryName             = GB
stateOrProvinceName     = England
localityName            = Camford
organizationName        = Camford University
emailAddress            = support@camford.ac.uk
commonName              = "Camford University FR Certificate Authority"
```

   b. In the `server.cnf` file:

      i. In the `[ req ]` section, add `encrypt_key = no`
      ii. In the `[CA_default]` section, change the default_days from 60 to a higher number (this is how long the certificates you create will be valid for). When the certificates expire, you will have to recreate them.
      iii. in the `[ server ]` section, change all of the parameters to match those of your organisation. e.g.

```
[server]
countryName             = GB
stateOrProvinceName     = England
localityName            = Camford
organizationName        = Camford University
emailAddress            = support@camford.ac.uk
commonName              = "Camford University FR Server Certificate"
```

When changing passwords in the `[ req ]` section of the server.cnf file, you must also update the `private_key_password` option in the FreeRADIUS `mods-available/eap` file with the same password.

We recommend that you do **not** change these defaults.

   c. In the `client.cnf` file:

      i. In the `[ req ]` section, add `encrypt_key = no`
      ii. In the `[CA_default]` section, change the default_days from 60 to a higher number (this is how long the certificates you create will be valid for). When the certificates expire, you will have to recreate them.
      iii. in the `[ client ]` section, change all of the parameters to match those of your organisation. e.g.

```
[client]
countryName             = GB
stateOrProvinceName     = England
localityName            = Camford
organizationName        = Camford University
emailAddress            = support@camford.ac.uk
commonName              = "Camford University FR Client Certificate"
```


All of the organisation parameters (`countryName`, `localityName`, etc) need to match in the three .cnf files but the `commonName` must be unique in each file)

3. Clear out any old certificates in the directory:

```
$ make destroycerts
```

4. Run the bootstrap script to generate the certificates

```
$ ./bootstrap
```

5. Create a file that is the concatenation of the certificate and private key of the client.
    a. Create the file

    ```
    $ openssl x509 -in client.crt > client.pem ; cat client.key >> client.pem
    ```

    b. Verify that the client.pem file starts with "`-----BEGIN CERTIFICATE-----`".

6. Because the above command was run as root, the keys and certificates created will not be readable by the FreeRADIUS user by default, and FreeRADIUS will not be able to start. To fix this, reset the group for the files:

```
$ chgrp freerad {client,server,ca,dh}*
```

3.1.2. Moonshot UI credential store

We need to enable the freeradius user to use the Moonshot UI flatstore:

```
$ echo "freerad" >> /etc/moonshot/flatstore-users
```

3.1.3. Set up the FreeRADIUS and Trust Router users

To allow FreeRADIUS to read a key database for dynamic realm support, we need to place the FreeRADIUS user and the Trust Router users into each other's groups to allow them to read shared files of each other.

```
$ adduser freerad trustrouter
$ adduser trustrouter freerad
```

3.1.4. RadSec

Next we need to configure RadSec. We do this by creating a file at `/etc/radsec.conf` with the following:

```
realm gss-eap {
        type = "TLS"
        cacertfile = "/etc/freeradius/certs/ca.pem"
        certfile = "/etc/freeradius/certs/client.pem"
        certkeyfile = "/etc/freeradius/certs/client.key"
        disable_hostname_check = yes
        server {
                hostname = "127.0.0.1"
                service = "2083"
                secret = "radsec"
        }
}
```

3.1.5. Dynamic Realm supprt

We next need to tell your FreeRADIUS server to support dynamic lookup of realms.

1. Open `/etc/freeradius/proxy.conf` for editing:
    a. Towards the top of the file is a stanza beginning "`proxy server {`". Find this.
    b. Below this, add `dynamic = yes`, like so:

```
proxy server {
        dynamic = yes
```

3.1.6. Channel Binding Support

We next need to configure your FreeRADIUS server to support channel bindings.

1. Open `/etc/freeradius/sites-available/abfab-tls` for editing:
    a. Scroll to the `client default` stanza at the bottom of the file
    b. Edit the stanza to match the below:

```
client default {
        ipaddr = 0.0.0.0/0
        proto = tls
        gss_acceptor_realm_name = "your RP realm here"
        trust_router_coi = ov-apc.moonshot.ja.net
}
```

> ⚠ **gss_acceptor_realm_name**
>
> For simple deployments, specify the same RP realm as in the `rp_realm` option in Section 4.1 below. For simple deployments, this usually matches your IDP Realm. For extended pilots or production environments, you should specify a realm value that will match all the hosts you will be connecting to your RP Proxy.
>
> Additionally, you must add a domain wildcard constraint in the Jisc Assent Portal that will match this realm value.

    c. If you have any other client definitions here, for example to distinguish between internal and external clients, also apply the change to them.

4. Configure the Trust Router Client

If you are going to connect your Moonshot RP Proxy to a Trust Router network, then the next step involves configuring the Trust Router client software and configuring its connection to a Trust Router.

4.1. Configure FreeRADIUS to use Trust Router

4.1.1. Configuring FreeRADIUS realm lookup

We need to configure the community and rp_realm appropriate for your Moonshot service, and the Trust Router that it will connect to.

1. Open the `/etc/freeradius/mods-enabled/realm` for editing.
2. Find the "realm suffix {" configuration directive, and fill out the fields as appropriate.

3. Repeat this for the "realm bangpath {" configuration directive.
4. For the default Jisc Assent Trust Router this will look like the following:

```
realm suffix {
  format = suffix
  delimiter = "@"
  default_community = "ov-apc.moonshot.ja.net"
  rp_realm = "Your service realm as registered in the Jisc Assent Portal"
  trust_router = "tr.moonshot.ja.net"
  rekey_enabled = yes
}

realm bangpath {
  format = prefix
  delimiter = "!"
  default_community = "ov-apc.moonshot.ja.net"
  rp_realm = "Your service realm as registered in the Jisc Assent Portal"
  trust_router = "tr.moonshot.ja.net"
  rekey_enabled = yes
}
```

Example

Camford University has a Moonshot service registered in the Jisc Assent Portal using the service realm of moonshot.camford.ac.uk, so its realm file would look like this:

```
realm suffix {
  format = suffix
  delimiter = "@"
  default_community = "ov-apc.moonshot.ja.net"
  rp_realm = "moonshot.camford.ac.uk"
  trust_router = "tr.moonshot.ja.net"
  rekey_enabled = yes
}

realm bangpath {
  format = prefix
  delimiter = "!"
  default_community = "ov-apc.moonshot.ja.net"
  rp_realm = "moonshot.camford.ac.uk"
  trust_router = "tr.moonshot.ja.net"
  rekey_enabled = yes
}
```

4.1.2. Register your Trust Router client with a Trust Router

At this point, the Moonshot service needs to be associated with a Trust Router. To do this, you need to contact the operator of a Trust Router you wish to join for their specific instructions on how to do this.

Once you have joined the Trust Router service, you will be issued with a Trust Router credential file in XML file format.

Keep this credential file safe. It usually will only be issued once and any subsequent requests usually invalidate any previously issued credentials. This is a security precaution.

Jisc Assent service instructions

The below instructions are specific to the world's first Trust Router service, Jisc Assent, operated by Jisc in the United Kingdom:

1. If you are not signed up to Assent, sign up to Assent first. This step may take a day or two while your organisation details are verified and you are invited to join the portal.
2. If you are signed up to Assent, log into the Assent portal.

   For more information about the Assent Portal, see the Assent Portal Primer.

3. Download a Trust Router credential under the 'Credential' section of your organisation in the portal (in the form of an XML file). Keep this file safe!

1. You must import the issued credential file using the `moonshot-webp` command as the `freerad` user:

```
$ su - --shell /bin/bash freerad
$ unset DISPLAY
$ moonshot-webp -f [path to credential file]
```

2. Check that the credential has been correctly imported:

```
$ ls -la /etc/freeradius/.local/share/moonshot-ui/identities.txt
```

3. If the file exists, the credential file's contents should be present in the file.

5. Testing

Now that we have the Moonshot RP Proxy installed and configured, we're now ready to test!

Tip

At this point you probably want two consoles open on the server, so that you can manually run various components separately.

5.1. Testing FreeRADIUS locally

The first test is to check whether FreeRADIUS is working in its most basic manner.

1. In window 1, run (as the freerad user)

```
$ su --shell /bin/bash freerad
$ unset DISPLAY
$ freeradius -fxx -l stdout
```

2. Check that no errors are output.

5.2. Testing the Trust Router connection

To test the connection to Trust Router, we need to make sure the Temporary Identity Server (TIDS) software is running, then use the Temporary Identity Client (TIDC) software to simulate a connection to the Trust Router.

5.2.1. Testing using the Temporary Identity Client (TIDC)

1. In window 2, (as the freerad user) run the tidc command:

```
$ su --shell /bin/bash freerad
$ unset DISPLAY
$ tidc tr.moonshot.ja.net [your rp-realm] ov-apc.moonshot.ja.net ov-apc.moonshot.ja.net
```

This uses the "tidc" binary which is used in the following way - *tidc [hostname-of-trust-router] [rp-realm] [hostname-of-apc-server] [apc-name]*

2. If the Trust Router connection was successful, you should see something like the following:

In window 2 - TIDC output

```
TIDC Client:
Server = tr.moonshot.ja.net, rp_realm = moonshot-rp.camford.ac.uk, target_realm = ov-apc.moonshot.ja.
net, community = ov-apc.moonshot.ja.net
connecting to host 'tr.moonshot.ja.net' on port 12309
CTRL-EVENT-EAP-STARTED EAP authentication started
CTRL-EVENT-EAP-PROPOSED-METHOD vendor=0 method=21
CTRL-EVENT-EAP-METHOD EAP vendor 0 method 21 (TTLS) selected
CTRL-EVENT-EAP-PEER-CERT [...]
CTRL-EVENT-EAP-SUCCESS EAP authentication completed successfully
tidc_fwd_request: Sending TID request:

[...]

tr_msg_decode_tidresp(): Success! result = success.
tr_msg_decode_servers(): Number of servers = 1.
Response received! Realm = ov-apc.moonshot.ja.net, Community = ov-apc.moonshot.ja.net.
Client Key Generated (len = 256):

[...]
```

6. Next Steps

At this point, you now have a Moonshot RP that is working and registered with a Trust Router. Now for the next steps:

6.1. Automatically start the software

6.1.1. FreeRADIUS

To automatically start FreeRADIUS, issue the following command (as root):

```
$ sudo update-rc.d freeradius defaults
```

6.2. Configure clients

The next step is to configure the Moonshot RP Proxy to Talk to Applications/Services.