Install an APC on Debian 7

On this page you will find instructions on how to set up a Trust Router APC on Debian 7 (Wheezy) using FreeRADIUS. It also installs and configures the Trust Router client.

Contents

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 7

The first thing that is required is a Debian 7 machine - this can be physical or virtual.

1. Install Debian 7 (Wheezy) 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 7

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 Wheezy 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 wheezy 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 Software

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 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 3.1 below.

3. Configure the Moonshot APC

Next, we need to configure the Moonshot APC.

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. 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"
        }
}
```

3.1.3. Realm

We next need to configure your realm in the FreeRADIUS server so that it knows not to send any requests for your own users off to another server.

1. Configure your realm in `/etc/freeradius/proxy.conf`
   a. Open the file for editing and find the line “realm example.com {“
   b. Above this, add the following, where YOUR_REALM should be substituted for the realm you intend to use for your APC:

```
realm YOUR_REALM {
        # Intentionally left blank
}
```

3.1.4. 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 APC realm here"
        trust_router_coi = "your APC realm here"
}
```

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

3.1.5. EAP Type

1. Set the EAP type in use by Moonshot (EAP-TTLS) by editing `/etc/freeradius/mods-enabled/eap`. Find the first instance of `default_eap_type = md5` and change it to TTLS, i.e.:

```
default_eap_type = ttls
```

Other EAP types should be supported (PEAP and MD5 have been tested).

3.1.6. Returning the User-Name

The APC must return the User-Name attribute in its RADIUS response.

1. As root, find the `post-auth` section in the `/etc/freeradius/sites-available/abfab-tr-idp` file.

   a. Insert the below at the top of the section, if it does not already exist:

```
        #
        #  For EAP-TTLS and PEAP, add the cached attributes to the reply.
        #  The "session-state" attributes are automatically cached when
        #  an Access-Challenge is sent, and automatically retrieved
        #  when an Access-Request is received.
        #
        #  The session-state attributes are automatically deleted after
        #  an Access-Reject or Access-Accept is sent.
        #
        update {
                &reply: += &session-state:
        }
```

   b. Save the file.
2. As root, find the `post-auth` section in the `/etc/freeradius/sites-available/inner-tunnel` file.
   a. At the top of the `post-auth` section, insert the following text:

```
        #
        #  Return the User-Name
        #
        update reply {
                User-Name := &request:User-Name
        }
```

b. Then look for the following text towards the bottom of the `post-auth` section:

```
        #
        #  Instead of "use_tunneled_reply", uncomment the
        #  next two "update" blocks.
        #
#        update {
#                &outer.session-state: += &reply:
#        }

        #
        #  These attributes are for the inner session only.
        #  They MUST NOT be sent in the outer reply.
        #
        #  If you uncomment the previous block and leave
        #  this one commented out, WiFi WILL NOT WORK,
        #  because the client will get two MS-MPPE-keys
        #
#        update outer.session-state {
#                MS-MPPE-Encryption-Policy !* ANY
#                MS-MPPE-Encryption-Types !* ANY
#                MS-MPPE-Send-Key !* ANY
#                MS-MPPE-Recv-Key !* ANY
#                Message-Authenticator !* ANY
#                EAP-Message !* ANY
#                Proxy-State !* ANY
#        }
```

c. If the text does not exist, insert it above the comments that describe the line "`Post-Auth-Type REJECT {`".
d. Remove the comments from the update statements in the text.
e. Save the file.

4. Resource Provider Authentication

All Resource Providers in the Trust Router network, including all IdPs and RP Proxies and the Trust Router itself, need to authenticate themselves to the APC using Moonshot. This means that for every service or organisation, you must provision a credential on the APC.

In a production environment, we recommend you use a method of Resource Provider Authentication that integrates well with your chosen method of managing your Trust Router infrastructure.

See Configuring FreeRADIUS to Use a Local Identity Store for options to define credentials.

We recommend using an automatic means to provision credential files, such as an online portal.

4.1. Defining the APC credential

During testing, we recommend using the FreeRADIUS users file to define credentials that your Resource Providers can use to authenticate to the APC. We will create a user with username "testapc" and password "testing".

1. Open `/etc/freeradius/users` for editing and put the following at the top of the file:

```
testapc        Cleartext-Password := "testing"
                        Reply-Message = "Hello test user. You have authenticated!"
```

The formatting of the stanza above is very important. There should be a <tab> in between the username and Cleartext-Password, and a line break followed by a <tab> before the Reply-Message.

4.2. Provisioning the APC credential

For the APC credential you defined in the previous step, create a Moonshot credential XML file:

1. Set the `<user>` tag to the credential you defined in the previous step, e.g. `testapc`
2. Set the `<password>` tag to its appropriate password. You may wish to base64-encode the password.
3. Set the `<realm>` tag to `YOUR_APC_REALM`.
4. You can leave the `<services>` tag out.
5. You should set the `<selection-rules>` tag to:

selection-rules

```
    <selection-rules>
       <rule>
         <pattern>trustidentity/*</pattern>
         <always-confirm>false</always-confirm>
       </rule>
    </selection-rules>
```

6. Define either of the two trust anchors as per the moonshot-webp XML Format documentation.

For simple test infrastructures, you may leave out the trust anchors, but it is not recommended.

7. Save the file, then deploy it onto the Trust Router that you are connecting to this APC (see Section 3.2.2 of Install a Trust Router).

5. Configure the Trust Router connection

The APC is fundamental to a Trust Router network, so the next step involves configuring the Trust Router client software and configuring its connection to a Trust Router.

5.1. Set up the FreeRADIUS and Trust Router users

We need to place the FreeRADIUS user and the Trust Router users into each other's groups to allow them to read each other's shared files.

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

5.2. Configure TIDS

The APC also runs the Temporary ID Server (TIDS).

1. Open the `/etc/default/trust_router` file for editing. If necessary, create it.

```
ipaddr="[your server IP]"
hostname="[your server hostname]"
gssname="testapc@YOUR_APC_REALM"

TIDS_USER="trustrouter"
TIDS_GROUP="trustrouter"
```

6. Testing

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

Tip

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

6.1. Testing FreeRADIUS locally

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

1. In window 1, run (as root user)

   ```
   $ freeradius -fxx -l stdout
   ```

2. In window 2, run (as root user)

   ```
   $ radtest testapc@YOUR-APC-REALM testing localhost 2222 testing123
   ```

   This uses the "radtest" utility which is used in the following way - *radtest username password servername port shared-secret*

3. If this is working correctly you should see something like the following:

   In window 1 - FreeRADIUS server output

   ```
   Sending Access-Accept of id 57 from 127.0.0.1 port 1812 to 127.0.0.1 port 33363
        Reply-Message = 'Hello test user. You have authenticated!'
   (1) Finished request 1.
   Waking up in 0.3 seconds.
   Waking up in 4.6 seconds.
   (1) Cleaning up request packet ID 57 with timestamp +94
   Ready to process requests.
   ```

   In window 2 - radtest client output

   ```
   Sending Access-Request of id 57 from 0.0.0.0 port 33363 to 127.0.0.1 port 1812
        User-Name = 'testapc'
        User-Password = 'testing'
        NAS-IP-Address = 127.0.1.1
        NAS-Port = 2222
        Message-Authenticator = 0x00
   rad_recv: Access-Accept packet from host 127.0.0.1 port 1812, id=57, length=61
        Reply-Message = 'Hello test user. You have authenticated!'
   ```

6.2. 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.

6.2.1. Starting the Temporary Identity Server (TIDS)

In window 3 (window 1 should still be still running the FreeRADIUS server and window 2 the radtest command), run the TIDS software:

```
$ tids [your server IP] testapc@YOUR-APC-REALM [your server hostname] /var/lib/trust_router/keys
```

testapc@YOUR-APC-REALM is the identity that the trust router will use when provisioning keys - this makes it easy to spot in your own log files. Specifying your server's IP *and* hostname may seem redundant (and for single server deployments, it is!). You'll need to set the hostname and IP arguments a little differently if you want to enable some more advanced configurations (such as load balancing and key sharing).

This uses the "tids" binary which is used in the following way - *tids [your-ip-address] trustrouter-gss-name] [your-hostname] [path-to-key-database]*

When using Network Address Translation (NAT) or a firewall, you must specify your external IP address.

6.2.2. Run an APC authentication test

At this point, you must configure your trust router to use testapc@YOUR-APC-REALM as authentication.

1. The trust router configuration must be updated with the test user associated with your trust router's rp_realm filter lines.
2. The trust router configuration must be updated with your new APC designated as the APC for your trust router.
3. The trust router must have its Moonshot credential store updated with the test user and its password.
4. The trust router must be restarted. At this point, the trust router will attempt to authenticate itself to the APC.
5. In the FreeRADIUS console, you should see an Access-Accept response.

7. Next Steps

At this point, you now have a Moonshot APC that is working. Now for the next steps:

7.1. Automatically start the software

7.1.1. FreeRADIUS

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

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

If this is working correctly, you should see FreeRADIUS running as a daemon process.

7.1.2. TIDS

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

```
$ sudo update-rc.d tids defaults
$ sudo service tids start
```

If this is working correctly, you should see TIDS running as a daemon process.

7.2. Configure a real source of Authentication

Your FreeRADIUS server can currently only authenticate a single user - "testapc". At this point, you will want to connect to your management database. The FreeRADIUS site has information and instructions for how to do this.