Table of Contents

Configuring FreeRADIUS as a balancing proxy

Assignment

Radius General Description.

When the client authenticates, the server checks if the username and password match the data from the user database. If the data matches, the client is authorized. However, if the number of users is large, the authorization process may take a long time.

To solve this problem it is proposed to use parallel servers and load balancing between them depending on the workload of each server. In this case, a single entry point for clients is preserved - the Radius server, which is actually replaced by a proxy-server (it only balances requests between other servers known to it). The proxy server remembers that if it has authorized user A on server B, the request will be directed primarily to the same server B.

The FreeRadius package is configured as a balancing proxy server that distributes requests from many clients to a single access point to multiple servers.

Configuring balancing of client requests between servers

Step 1. Add the proxy to the list of server clients

Add a proxy definition as a client to the end of the `clients.conf` file of the **servers** configuration:

```
client client1 {
    ipv4addr = 10.10.10.61
    secret = testing123
}
```

Step 2. Add the client to the proxy client list

Add a client definition to the end of the `clients.conf` configuration **proxy** file:

```
client client1 {
    ipv4addr = 10.10.10.60
    secret = testing123
}
```

Step 3. Add servers and balancing parameters to the proxy configuration

In the proxy.conf file of the proxy configuration:

1. Duplicate the `home_server localhost {}` section accordingly to the number of servers;
2. Rename each new section and customize the `ipv4addr = server_address` parameter:

```
home_server server1 {
```

```
    ipv4addr = 10.10.10.62
}
home_server server2 {
    ipv4addr = 10.10.10.63
}
```

3. In the `home_server_pool my_auth_failover {}` section
    1. Change balancing mode - change parameter `type = fail-over` to `type = load-balance` (description of balancing modes in the Balancing Mode Options section)
    2. Comment out the line `home_server = localhost`.
    3. For each server, add the lines `home_server = section_name` from item 2

```
home_server_pool my_auth_failover {
    type = load-balance
    //home_server = localhost
    home_server = server1
    home_server = server2
}
```

Optionally: to improve performance, you can disable unnecessary validation modules in the `mods-enabled` directory (there are symbolic links to module configurations that "must be loaded").

Balance mode options

1. **`fail-over`** - the request is sent to the first live home server in the list. i.e. If the first home server is marked "dead", the second one is chosen, etc.
2. **`load-balance`** - the least busy home server is chosen, where "least busy" is counted by taking the number of requests sent to that home server, and subtracting the number of responses received from that home server.

    If there are two or more servers with the same low load, then one of those servers is chosen at random. This configuration is most similar to the old "round-robin" method, though it is not exactly the same.

    Note that load balancing does not work well with EAP, as EAP requires packets for an EAP conversation to be sent to the same home server. The load balancing method does not keep state in between packets, meaning that EAP packets for the same conversation may be sent to different home servers. This will prevent EAP from working.

    For non-EAP authentication methods, and for accounting packets, we recommend using "load-balance". It will ensure the highest availability for your network.

3. **`client-balance`** - the home server is chosen by hashing the source IP address of the packet. If that home server is down, the next one in the list is used, just as with "fail-over".

    There is no way of predicting which source IP will map to which home server.

This configuration is most useful to do simple load balancing for EAP sessions, as the EAP session will always be sent to the same home server.

4. **`client-port-balance`** - the home server is chosen by hashing the source IP address and source port of the packet. If that home server is down, the next one in the list is used, just as with "fail-over".

   This method provides slightly better load balancing for EAP sessions than "client-balance". However, it also means that authentication and accounting packets for the same session MAY go to different home servers.

5. **`keyed-balance`** - the home server is chosen by hashing (FNV) the contents of the Load-Balance-Key attribute from the control items. The request is then sent to home server chosen by taking:

   ```
   server = (hash % num_servers_in_pool)
   ```

   If there is no Load-Balance-Key in the control items, the load balancing method is identical to "load-balance".

   For most non-EAP authentication methods, The User-Name attribute provides a good key. An "unlang" policy can be used to copy the User-Name to the Load-Balance-Key attribute. This method may not work for EAP sessions, as the User-Name outside of the TLS tunnel is often static, e.g. "anonymous@realm".