Содержание

Настройка FreeRADIUS как балансирующего прокси 3
Назначение 3
Настройка балансировки запросов клиентов между серверами 3
Варианты режимов балансировки 4

Настройка FreeRADIUS как балансирующего прокси

Назначение

Пакет FreeRadius настроен как балансирующий proxy-сервер, который распределяет запросы от многих клиентов на одну точку доступа к нескольким серверам.

Когда клиент проходит аутентификацию на RADIUS сервере оператора, на сервере происходит проверка соответствия имени пользователя и пароля данным из базы данных пользователей. Если данные совпадают, то клиент проходит авторизацию. Однако при большом количестве пользователей процесс авторизации может занимать длительное время.

Для решения этой проблемы предлагается использовать параллельные серверы и балансировку нагрузки между ними в зависимости от загруженности каждого сервера. При этом для fastPCRF сохраняется единая точка входа — сервер FreeRADIUS, который выступает как proxy-сервер (он только балансирует запросы между известными ему другими операторскими RADIUS-серверами). Proxy-сервер запоминает, что если он авторизовал пользователя A на сервере B, то запрос будет направлен преимущественно на этот же сервер B.

Настройка балансировки запросов клиентов между серверами

Шаг 1. Добавление прокси в список клиентов серверов

В конфигурационном файле `clients.conf` на стороне **Radius-сервера** определить прокси как клиента:

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

Шаг 2. Добавление клиента в список клиентов прокси

В конфигурационном файле `clients.conf` на стороне **прокси** определить клиента:

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

Шаг 3. Добавление серверов и параметров балансировки в конфигурацию прокси

В файле `proxy.conf` конфигурации прокси:

1. Продублировать секцию `home_server localhost {}` соответственно числу серверов;
2. Переименовать каждую новую секцию и настроить параметр `ipv4addr =` адрес_сервера:

```
home_server server1 {
    ipv4addr = 10.10.10.62
}
home_server server2 {
    ipv4addr = 10.10.10.63
}
```

3. Отредактировать раздел `home_server_pool my_auth_failover {}`
    1. Изменить режим балансировки — изменить параметр `type = fail-over` на `type = load-balance` (описание режимов балансировки в разделе Варианты режимов балансировки)
    2. Закомментировать строчку `home_server = localhost`
    3. Для каждого сервера добавить строки `home_server =` имя_секции из пункта 2

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

Опционально: для повышения быстродействия можно отключить лишние модули проверки в каталоге `mods-enabled` (там находятся символические ссылки на конфигурации модулей, которые "должны быть загружены").

Варианты режимов балансировки

1. **`fail-over`** — запрос отправляется на первый живой домашнему серверу в списке. Т.е. если первый домашний сервер отмечен как "мертвый", то выбирается второй и т.д.

2. **`load-balance`** — выбирается наименее загруженный домашний сервер, где "наименьшая загруженность" определяется путем определения количества запросов, отправленных на этот домашний сервер, и вычитанием количество ответов, полученных от этого домашнего сервера.

   Если существует два или более сервера с одинаково низкой нагрузкой, то один из них выбирается случайным образом. Такая конфигурация наиболее похожа на старый "round-robin", хотя это не совсем так.

   Обратите внимание, что балансировка нагрузки не очень хорошо работает с EAP, поскольку EAP требует, чтобы пакеты для EAP-взаимодействия отправляться на один и тот же домашний сервер. Метод балансировки нагрузки Не сохраняет состояние между

пакетами, что означает, что EAP-пакеты для одного и того же разговора могут быть отправлены на разные домашние серверы. Это не позволит EAP работать.

Для методов аутентификации, отличных от EAP, и для учетных пакетов мы рекомендуем использовать "load-balance". Это позволит обеспечить максимальную доступность вашей сети.

3. **`client-balance`** — домашний сервер выбирается путем хэширования IP-адреса источника пакета. Если этот домашний сервер не работает, то используется следующий по списку, как и в случае при "fail-over".

   Невозможно предсказать, какой IP-адрес источника будет сопоставлен к какому домашнему серверу.

   Эта конфигурация наиболее полезна для выполнения простой балансировки нагрузки балансировки нагрузки для сеансов EAP, поскольку сеанс EAP будет всегда будет отправляться на один и тот же домашний сервер.

4. **`client-port-balance`** — выбор домашнего сервера осуществляется путем хэширования IP-адреса и порта источника пакета. Если этот домашний сервер не работает, то используется следующий по списку используется, как и в случае с "fail-over".

   Этот метод обеспечивает несколько лучшую балансировку нагрузки для сессий EAP, чем "client-balance". Однако это также означает, что пакеты аутентификации и учета для одного и того же сеанса МОГУТ отправляться на разные домашние серверы.

5. **`keyed-balance`** — выбор домашнего сервера осуществляется путем хэширования (FNV) содержимого атрибута Load-Balance-Key из элементов управления. Затем запрос отправляется на домашний сервер выбранный пользователем:

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

   Если в элементах управления отсутствует Load-Balance-Key, метод балансировки нагрузки идентичен "load-balance".

   Для большинства не-EAP методов аутентификации атрибут User-Name является хорошим ключом. Политика "unlang" может быть использована для копирования User-Name в атрибут Load-Balance-Key атрибут. Этот метод может не работать для сессий EAP, поскольку имя пользователя вне туннеля TLS часто является статическим статичным, например, "anonymous@realm".