**Сертификаты на два сайта:**

***В конфиге haproxy***

#==================FRONTEND BLOCK========================================

#=====letsencrypt

frontend letsencrypt

bind \*:80

bind \*:443 ssl crt /etc/ssl/virus-cmv.ru/virus-cmv.ru.pem

bind \*:443 ssl crt /etc/ssl/virus-hpv.ru/virus-hpv.ru.pem

# Let's Encrypt certbot path

acl letsencrypt-acl path\_beg /.well-known/acme-challenge/

use\_backend letsencrypt if letsencrypt-acl

# Redirect if HTTPS is \*not\* used

redirect scheme https code 301 if !{ ssl\_fc }

default\_backend my\_http\_back

#=====letsencrypt

frontend stats

bind \*:8404

stats enable

stats uri /stats

stats refresh 10s

#listen ssl

# bind \*:443

# mode tcp

# option tcplog

# balance roundrobin

# server server1 192.168.0.34:443

# server server2 192.168.0.35:443

listen ftp

bind :21

bind :65523-65535

mode tcp

server ftp-server 192.168.0.25 check port 21

listen ssh

bind :22222

mode tcp

server ssh-server-jenkins-two-site 192.168.0.35 check port 22222

#frontend my\_ssl

# bind 195.178.199.99:443

# mode tcp

# default\_backend my\_ssl

#frontend my\_http\_front

# bind 195.178.199.99:80

# default\_backend my\_http\_back

#frontend my\_https\_front

# option tcplog

# mode tcp

# bind 195.178.199.99:443

# default\_backend my\_https\_back

frontend my\_secret\_front

bind \*:8888

default\_backend my\_secret\_back

frontend my\_jenkins\_front

bind \*:8080

default\_backend my\_jenkins\_back

frontend my\_zabbix\_front

bind 195.178.199.99:20050

default\_backend my\_zabbix\_back

frontend my\_openmediavault

bind 195.178.199.99:5757

default\_backend my\_openmediavault

#====================BACKEND BLOCK===================================

#=====letsencrypt

backend letsencrypt

# mode http

server letsencrypt 127.0.0.1:9999

#=====letsencrypt

backend my\_http\_back

balance roundrobin

server myweb2 192.168.0.34:80 weight 50

server myweb1 192.168.0.35:80 weight 50

#backend my\_ssl

# balance roundrobin

# mode tcp

# server myweb2 192.168.0.34:443 weight 50

# server myweb1 192.168.0.35:443 weight 50

#backend my\_https\_back

# balance roundrobin

# mode tcp

# option ssl-hello-chk

# server ssl 192.168.0.35:443 weight 50

# server ss2 192.168.0.34:443 weight 50

backend my\_secret\_back

server mysecretweb1 192.168.0.35:8888

server mysecretweb2 192.168.0.34:8888

backend my\_jenkins\_back

server jenkins 192.168.0.40:8080

backend my\_zabbix\_back

server zabbix 192.168.0.36:20050

backend my\_openmediavault

server openmediavault 192.168.0.25:80

#==========================================================

***https://serversforhackers.com/c/letsencrypt-with-haproxy***

LetsEncrypt с HAProxy

Мы рассмотрим использование LetsEncrypt для создания сертификатов SSL с балансировщиком нагрузки HAProxy.

Прокси

[LetsEncrypt остроумие ...](http://serversforhackers.com/c/letsencrypt-with-haproxy)

^ Рекламное пространство, чтобы помочь компенсировать расходы на хостинг: D

**LetsEncrypt с HAProxy**

Это видео из модуля балансировки нагрузки курса [Scaling Laravel](https://serversforhackers.com/scaling-laravel) .

Частично я хотел рассказать о том, как использовать SSL-сертификаты с балансировщиком нагрузки HAProxy. LetsEncrypt (certbot) отлично подходит для этого, поскольку мы можем получить бесплатный и надежный сертификат SSL. Поскольку мы используем LetsEncrypt на балансировщике нагрузки (HAProxy), который не может обслуживать HTTP-запросы авторизации, которые делает LetsEncrypt, у нас есть некоторые уникальные проблемы, которые необходимо решить. Посмотрим как!

**Установить LetsEncrypt**

Давайте уберемся от шаблона. Вот как я устанавливаю LetsEncrypt (Certbot) в Ubuntu 16.04:

sudo add-apt-repository -y ppa:certbot/certbot

sudo apt-get update

sudo apt-get install -y certbot

Как показано на видео, этот установщик создает задачу CRON ( /etc/cron.d/certbot) для запроса продления два раза в день. Сертификат обновляется только в том случае, если до истечения 30 дней. Проверка дважды в день - относительно безопасный способ проверить и обойти возможные ошибки синхронизации. Это значение по умолчанию очень удобно для типичной установки.

Однако, поскольку у нас есть некоторые уникальные потребности в HAProxy, мы будем использовать немного другую задачу CRON для этого варианта использования.

**Проблемы**

Первое препятствие, которое необходимо обойти, возникает из-за того, что LetsEncrypt авторизует сертификат для сервера, запрашивая файл через HTTP (S) запрос. Однако HAProxy *не* является веб-сервером. Он не будет обслуживать файлы сам по себе - он только перенаправит запрос в другое место. Наши серверы приложений не смогут обработать этот запрос авторизации.

Поскольку нам нужен наш SSL-сертификат на балансировщике нагрузки (SSL Termination), наша цель - найти способ, чтобы HAProxy распознал запрос от LetsEncrypt и направил его в веб-службу, которая ответит ответом, который LetsEncrypt необходим для авторизации сертификата.

LetsEncrypt поставляется с собственным встроенным слушателем веб-сервера для такого случая использования, поэтому мы можем это сделать!

Второе препятствие заключается в том, что HAProxy ожидает, что все сертификаты SSL будут в одном файле, который включает цепочку сертификатов, корневой сертификат и закрытый ключ. HAProxy имеет закрытый ключ в отдельном файле, поэтому наш последний шаг - объединить файлы во что-то, что HAProxy может прочитать.

Наконец, мы также решим проблему автоматизации продлений с учетом вышеуказанных ограничений.

**Рабочий процесс**

Мы просим LetsEncrypt о двух действиях:

1. Запросить новый сертификат
2. Продлить существующий сертификат

После каждого шага, описанного выше, нам также необходимо объединить полученные файлы сертификатов в формат, который хочет HAProxy.

Давайте посмотрим на эти два сценария, а затем посмотрим, как объединить сертификат в один файл.

**Настройка HAProxy**

Когда мы запрашиваем новый сертификат, LetsEncrypt запрашивает файл авторизации (например, URI /.well-known/acme-challenge/random-hash-here). Этот запрос будет выполняться через порт 80, поскольку предположительно еще не настроен сертификат.

Интересно, что если HAProxy прослушивает порт 443, LetsEncrypt может попытаться авторизоваться через него. Итак, когда мы создаем новый сертификат, нам нужно, чтобы HAProxy прослушивал только порт 80 .

Еще одна проблема: HAProxy прослушивает порт 80. Однако нам понадобится LetsEncrypt, чтобы настроить его автономный сервер для прослушивания запросов авторизации. По умолчанию он также будет использовать порт 80, что вызовет конфликт, поскольку только один процесс может прослушивать порт одновременно. Итак, нам нужно указать LetsEncrypt, что нужно прослушивать другой порт!

В HAProxy мы можем спросить, содержит ли входящий HTTP-запрос строку /.well-known/acme-challenge. В приведенной ниже конфигурации, если HAProxy видит, что запрос действительно включает этот URI, он направляет запрос в LetsEncrypt. В противном случае он будет перенаправлять запрос на любые серверы в ротации балансировщика нагрузки как обычно.

# The frontend only listens on port 80

# If it detects a LetsEncrypt request, is uses the LE backend

# Else it goes to the default backend for the web servers

frontend fe-scalinglaravel

bind \*:80

# Test URI to see if its a letsencrypt request

acl letsencrypt-acl path\_beg /.well-known/acme-challenge/

use\_backend letsencrypt-backend if letsencrypt-acl

default\_backend be-scalinglaravel

# LE Backend

backend letsencrypt-backend

server letsencrypt 127.0.0.1:8888

# Normal (default) Backend

# for web app servers

backend be-scalinglaravel

# Config omitted here

Как только это настроено в HAProxy, мы можем перезагрузить его ( sudo service haproxy reload), а затем перейти к запуску LetsEncrypt.

**Новые сертификаты**

Команда для получения нового сертификата от LetsEncrypt, которую мы будем использовать, выглядит так:

sudo certbot certonly --standalone -d demo.scalinglaravel.com \

--non-interactive --agree-tos --email admin@example.com \

--http-01-port=8888

Давайте рассмотрим, что это делает:

1. --standalone - Создайте автономный веб-сервер для прослушивания HTTP-запроса авторизации сертификата.
2. -d demo.scalinglaravel.com- Домен, для которого создается сертификат. Вы можете использовать несколько -dфлагов для нескольких доменов для одного сертификата. Домены должны маршрутизироваться на сервер, для которого мы создаем сертификат (для домена должен быть настроен DNS).
3. --non-interactive --agree-tos --email admin@example.com - Сделайте это неинтерактивным, сказав это, согласившись с УО и сообщив LetsEncrypt электронное письмо, которое будет использоваться для отправки уведомлений «ВАШ СЕРТ истекает».
4. --http-01-port=8888- Магия ™. Это указывает автономному серверу прослушивать порт 8888. Обратите внимание, что LetsEncrypt по- *прежнему* будет отправлять HTTP-запрос авторизации через порт 80. Однако прослушиватель ожидает, что прокси (например, наш сервер HAProxy) направит запрос к нему через порт 8888. Флаг означает, http-01что он ожидает HTTPзапрос, а НЕ HTTPSзапрос.

**Продление сертификатов**

Если мы обновляем сертификат, это, скорее всего, означает, что используется действующий сертификат HTTPS. Нам просто нужно, чтобы LetsEncrypt проделал тот же процесс, что и выше, чтобы обновить его. Однако есть несколько ключевых отличий:

1. Предположительно, HAProxy прослушивает порт 443 для SSL-соединений, а LetsEncrypt будет отправлять запрос авторизации по HTTPS вместо HTTP.
2. Автономный сервер будет ожидать от него HTTPS (технически TLS) запроса вместо обычного HTTP-запроса.

Итак, настройка HAProxy будет почти такой же, за исключением того, что на этот раз он будет прослушивать порт 443.

Мы рассмотрим настройку конфигурации HAProxy для SSL чуть позже.

frontend fe-scalinglaravel

bind \*:80

# This is our new config that listens on port 443 for SSL connections

bind \*:443 ssl crt /etc/ssl/demo.scalinglaravel.com/demo.scalinglaravel.com.pem

# New line to test URI to see if its a letsencrypt request

acl letsencrypt-acl path\_beg /.well-known/acme-challenge/

use\_backend letsencrypt-backend if letsencrypt-acl

default\_backend be-scalinglaravel

# LE Backend

backend letsencrypt-backend

server letsencrypt 127.0.0.1:8888

# Normal (default) Backend

# for web app servers

backend be-scalinglaravel

# Config omitted here

Обратите внимание, что автономный сервер LetsEncrypt все *еще* прослушивает порт 8888, хотя и ожидает TLS-соединение. Ничего страшного, номер порта на самом деле не имеет значения. Единственное изменение здесь заключается в том, что HAProxy также прослушивает SSL-соединения.

Вот как продлить сертификат с LetsEncrypt:

sudo certbot renew --tls-sni-01-port=8888

Вот и все! Мы используем renew, но на этот раз мы говорим ему ожидать tlsсоединения и продолжать прослушивание порта 8888 (снова).

**Сертификаты SSL и HAProxy**

Для HAProxy необходимо, чтобы ssl-сертификат был одним файлом в определенном формате. Для этого мы создаем новый каталог, в котором будет жить сертификат SSL, который читает HAProxy. Затем мы выводим «живые» (последние) сертификаты из LetsEncrypt и выгружаем этот вывод в файл сертификата для использования HAProxy:

sudo mkdir -p /etc/ssl/demo.scalinglaravel.com

sudo cat /etc/letsencrypt/live/demo.scalinglaravel.com/fullchain.pem \

/etc/letsencrypt/live/demo.scalinglaravel.com/privkey.pem \

| sudo tee /etc/ssl/demo.scalinglaravel.com/demo.scalinglaravel.com.pem

/etc/letsencrypt/live/your-domain-here.tldКаталог будет содержать символьные ссылки на свой ток, большинство сертификата уточненных.

Итак, мы проверяем, существует ли каталог для нашего сертификата, а затем объединяем содержимое fullchain.pemфайла (сертификат и цепочку сертификатов) и privkey.pemфайл закрытого ключа . Мы помещаем результаты в файл demo.scalinglaravel.com.pem. Порядок, в котором мы объединяем файлы, имеет значение (полная цепочка, за которой следует закрытый ключ).

Как мы видели, конфигурация HAProxy использует этот новый файл:

frontend fe-scalinglaravel

bind \*:80

# This is our new config that listens on port 443 for SSL connections

bind \*:443 ssl crt /etc/ssl/demo.scalinglaravel.com/demo.scalinglaravel.com.pem

# omitting the rest of the config...

**Автоматизация продления**

Чтобы автоматизировать продление нашего сертификата, нам нужно повторить вышеуказанные шаги:

1. Получите новый сертификат
2. Создайте новый файл сертификата для использования HAProxy.

По умолчанию LetsEncrypt создает запись CRON в /etc/cron.d/certbot. Запись выполняется дважды в день (по умолчанию LetsEncrypt обновляет сертификат только в том случае, если срок его действия истекает в течение 30 дней).

Что мне нравится делать, так это запускать bash-скрипт, который запускается ежемесячно, и каждый раз принудительно обновлять сертификат.

Мы можем начать с редактирования файла CRON, чтобы запускать скрипт ежемесячно:

0 0 1 \* \* root bash /opt/update-certs.sh

Это выполняется в нулевую минуту нулевого часа (полночь в любом часовом поясе, установленном на вашем сервере, вероятно, в формате UTC) в первый день каждого месяца.

Файл bash, указанный в задаче CRON ( /opt/update-certs.sh), выглядит так:

#!/usr/bin/env bash

# Renew the certificate

certbot renew --force-renewal --tls-sni-01-port=8888

# Concatenate new cert files, with less output (avoiding the use tee and its output to stdout)

bash -c "cat /etc/letsencrypt/live/demo.scalinglaravel.com/fullchain.pem /etc/letsencrypt/live/demo.scalinglaravel.com/privkey.pem > /etc/ssl/demo.scalinglaravel.com/demo.scalinglaravel.com.pem"

# Reload HAProxy

service haproxy reload

Это выполняет все шаги, которые мы выполняли ранее. Единственная разница в том, что я использую --force-renewalLetsEncrypt для ежемесячного обновления сертификата. Этот способ немного проще в рассуждении, и он не станет жертвой потенциальных ошибок синхронизации, которые пытались обойти дважды в день.

**Обеспечение HTTPS**

Это не связано с LetsEncrypt, а скорее с вашей реализацией SSL.

Если вы хотите принудительно использовать SSL в HAProxy, вы также можете сделать это, не влияя на способность LetsEncrypt продлевать сертификат:

frontend fe-scalinglaravel

bind \*:80

bind \*:443 ssl crt /etc/ssl/demo.scalinglaravel.com/demo.scalinglaravel.com.pem

# Redirect if HTTPS is \*not\* used

redirect scheme https code 301 if !{ ssl\_fc }

acl letsencrypt-acl path\_beg /.well-known/acme-challenge/

use\_backend letsencrypt-backend if letsencrypt-acl

default\_backend be-scalinglaravel

Это означает, что если внешнее соединение не использовало SSL, то верните перенаправление 301 на тот же URI, но с https.