Setup OPNsense with HAProxy and Let's Encrypt

Posted Sep 11, 2023 • Updated Apr 3, 2025

Create a reverse proxy with OPNsense and HAProxy using Let's Encrypt certificates

By Marcus Holtz

27 min read

Setting up HAProxy and Let's Encrypt on OPNsense

If you're reading this, wondering why my blog came up before the official documentation - they've removed the original documentation on account of thier enterprise-only Reverse Proxy and Webserver.

You can find a copy of the original documentation at archive.org

---

HAProxy uses ACME Let's Encrypt for SSL authentication

OPNSense's HAProxy package can use ACME for certificates.

First, we must install those two packages.

Installing required software

1. In your OPNsense, go to: `System --> Firmware --> Updates` and install all updates.

2. Then, head to: `System --> Firmware --> Plugins` and install the following plugins: `os-acme-client`, `os-haproxy`

There are several changes we have to make to the defaults of OPNsense before we can intake traffic to our router.

System preparation

In OPNsense go to: `System --> Settings --> Administration`

You will need to checkbox the `Disable web GUI redirect rule` and change the `Web GUI TCP port` to a number you can remember, example: 4443.

This change is to allow your router to reply to requests on the default ports for HAProxy's traffic (80/443). By moving the port number OPNsense uses, you're able to free up those port numbers for HAProxy.

Again, be sure you have change the `Web GUI TCP port` to ** a number you can remember. **

Then, reconnect to OPNsense's web interface using the port you entered, example: `https://192.168.1.1:4443`

Setting a Virtual IP

I like to use a virtual IP instead of just pointing all traffic back to the localhost.

It helps me see traffic a little bit better. If you want to create a virtual IP open: `Interfaces --> Virtual IPs --> Settings`

Creating a Virtual IP

When thinking about this new IP address for your "SSL Offloading Server" you would want to chose an IP that belongs to another network than any you're using.

The localhost subnet should, generally, be available to us to help avoid IP conflicts in your local network.

The chosen IP/Subnet will be the IP on which the HTTP_frontend and HTTPS_frontend will be listening on.

We can use anything from 127.0.0.0-127.255.255.255. For this, we will choose to use: `127.1.2.3/32`

1. `Mode` should be `IP Alias`
2. `Interface` can remain `LAN`
3. `Network Address` is the address we picked above `127.1.2.3/32`
4. Save.
5. Apply.

Port Alias for HAProxy's Ports

To make things easier on us, we should create an alias for all the ports we need HAProxy to use.

Go to: `Firewall --> Aliases`

Now, we are going to create an alias for the ports that HAProxy will be listening on. Click the add button to begin.

1. `Name` anything you like, example: `HAProxy_Ports`.
2. `Type` should be `Port(s)`
3. `Content` needs to be `80` and `443`. You can type the number and hit enter to save the field.
4. Now, both ports are set and we have a recognizable name for the alias, click save.

Let the traffic start following

Modify the firewall: `Firewall --> Rules --> WAN`

Make a firewall rule to allow any inbound traffic on the WAN interface connecting to the `HAProxy_Ports` alias.

1. Add a new firewall rule.
2. Set the `action` to `Pass`
3. `Interface` is `WAN`
4. `Direction` is `IN`
5. `TCP/IP` version `4`
6. `Protocol` needs `TCP`
7. `Source` can be `Any`
8. `Destination` needs to be `This Firewall`
9. `Destination port range` should be set to our `HAProxy_Ports` alias.
10. Save.

Appeasing the OCSP Must Staple

Head to settings: `System --> Settings --> Cron` to create a new cron job.

Because our certificate has the `OCSP Must Staple` extension we need to update HAProxy's OCSP data *regularly*.

If this isnt done, clients connecting to HAProxy will get a security warning and won't be able to connect.

The fix is to create a cron job

I have had issues with OCSP Staples so I set my cron job to run everyday, every hour. This is more than likly not nessasary and may be a part of this tutorial that I update… but until then…

1. Add a new cron job, set `any number` for `Minutes` and the other fields should be an `*`.
2. For `command` choose `Update HAProxy OCSP data`.
3. Save.

in HAProxy. This is because the ACME plugin restarts HAProxy after installing the new certificates.

Let's Encrypt (ACME Client)

Enable and Update Schedule

Moving on, go to: `Services --> ACME Client --> Settings`

1. Uncheck `Show introduction pages`
2. Check `Enable Plugin`
3. We don't need the `HAProxy integration`. Leave this one alone. It is for 'HTTP-01' and this tutorial is using the 'DNS-01' challenge.

Update Schedule

Click on the `Update Schedule` tab next to the Settings tab in the ACME Client services section.

This is located in: `Services --> ACME Client --> Settings --> Update Schedule`

- The `Update Schedule` tab will allows us to configure at which time of the day our certificates are renewed.

Why change time of day?

If everyone renews their certs at the same time, let's say, on the :00 of the hour – there will be a heavy load of renewals on the CA.

You want your renewal request to happen at a time of the day where there is not much load on your services as well. The ACME plugin restarts HAProxy so it can use your new certificates which results in a very short downtime of HAProxy.

- Pick from the numbers below for the `minutes` of the Renew ACME certificate command, the hour is up to your use case.

2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59

Restart services when there are changes

Once you complete the certificate renewal, you have to make sure the service restarts so it uses the new file.

To automate our service restart, visit: `Services --> ACME Client --> Automations`

Inorder to restart HAProxy after the cert updates, create a new automation.

1. `Name` can be anything to identify the automation, example: `RestartHAproxy`
2. `Run command` should select the command `Restart HAProxy (OPNsense plugin)`

Account and CA

Next, go to: `Services --> ACME Client --> Accounts`

1. On this page, enter a `Name` – for reference, I use the domain name (ssl.test.house.lan).

and have to wait a week to try again.

Note: You can use the staging environment, `Let's Encrypt Test CA`, instead of the default `ACME CA` `Let's Encrypt`

ACME DNS-01 challenge

To request a certificate, we need to issue a challenge.

Head to: `Services --> ACME Client --> Challenge Types`

To get a wildcard certificate we need to use a DNS challenge. This tells Let's Encrypt we own the **entire** domain and can therefore issue certificates to the subdomains beneath it.

1. For `Name` I would put the assocaited domain and the challenge type (acmedns.test.house.lan).
2. `Challenge Type` should be set to `DNS-01`
3. `DNS Service` is up to your provider.
   - I use CloudFlare, as anyone can use them by changing Nameservers. Create a new API Token (with correct credentials).
4. Save and double check you got the **Account ID, API Token, and all credential's permissions** set correctly.

Issuing a new certificate

Issue a certificate: `Services --> ACME Client --> Certificates`

1. `Common Name` is the URL of the certificate you're requesting. We want a wildcard for a subdomain (*.test.house.lan).
2. Select the `ACME Account` we created earlier (example: ssl.test.house.lan).
3. Choose the `Challenge Type` that you named above (example - acmedns.test.house.lan).
4. `Key Length` is a preference. I use `ec-384`. Generally, the higher number the better.
5. **Be sure** you check the `OSCP Must Staple` checkbox. This is a modern requirement.
6. `Automations` should have our restart policy we made earlier, select that now.
7. Save

> Depending on what Certificate Authority you chose earlier (Test CA or Not) will issue your certificate.

To issue your certificate, you need to hit the `circular arrow button`, it's alongside all the other buttons under `commands`

Log file view of the action

Check what happened at: `Services --> ACME Client --> Log Files --> ACME Log`

There are two tabs, **System** and **ACME Log**.

- **System** will display what OPNsense decided to do.
- **ACME Log** is the script that runs. You can find most errors here.

What if I used the Testing CA?

Go back to: `Services --> ACME Client --> Accounts`

Since you, presumably, successfully issued a staging certificate you can now change from the test environment to the default `ACME CA` `Let's Encrypt` and issue the production certificate.

Issue a production certificate

Back to: `Services --> ACME Client --> Certificates`

Now, again, you dont need to do this if you've already used the production server to generate a certificate.

Otherwise, forcefully issue your certificate with the `circular arrow button`

This time it you should receive a valid and trusted SSL certificate. Make sure to check the ACME log for any errors though!

HAProxy configuration

This is where the convoluted part begins. The instructions sound similar and become more difficult to understand. There are several sections, including using a map file, rules, conditions, backend server pools, services, dns overrides and more.

Please re-read if needed and be patient.

> Please note, a lot of these pages are located as **dropdowns** in the menu headers on the **settings** page.

HAProxy inital configuration

HAProxy Service

First, we need to set the defaults we're going to use across the HAProxy service in:

`Services --> HAProxy --> Settings --> Service`

On this page,

1. Uncheck `Show introduction pages`
2. Do checkbox `Store OCSP responses`

Enable HAProxy

Let's turn it on: `Services --> HAProxy --> Settings --> Service`

On this page just checkbox `Enable HAProxy` and hit `Apply.

1. Then, Apply.

The next drop down we need is: `Services --> HAProxy --> Settings --> Global Parameters`

1. In the upper left hand corner, check the `advanced mode` button.
2. The number of `HAProxy threads` should not exceed the number of CPU threads of your OPNsense (example, 4).
3. `Maximum connections` keeps from overloading the server, say `800`
4. Increase the `Maximum SSL DH Size` to `4096`
5. Apply.

HAProxy Default Paramaters

This area will helps us temper our reverse proxy to ensure there isnt an overloaded server.

Enter: `Services --> HAProxy --> Settings --> Default Parameters`

1. `Maximum Connections (Public Services)` set to `1200`
2. `Maximum Connections (Servers)` set to `8000`
3. You can leave everything else the same.
4. Apply.

HAProxy background services configuration

Setting conditions for non-HTTPS connections

Here we only need to create a NoSSL_condition, which is necessary in order to identify non-HTTPS traffic. This will be our only condition.

Next, go to: `Services --> HAProxy --> Settings --> Rules & Checks --> Conditions`

1. `Name` should be set to `NoSSL_condition`.
2. `Condition type` needs to be configured to `Traffic is SSL (locally deciphered)`.
3. `Negate condition` has to be checked for this rule to make sense.
4. Apply.

Map file for ease and convenience

Map files are found in: `Services --> HAProxy --> Settings --> Advanced --> Map Files`

Here we will create a new map file PUBLIC_SUBDOMAINS_map file for our public subdomains that we want to access from outside of our network.

our NEXTCLOUD_server").

1. `Name` is `PUBLIC_SUBDOMAINS_mapfile`
2. `Content` for this example, should look like:

```
# comment to tell you comments are allowed
nextcloud NEXTCLOUD_backend
```

3. Apply.

Rules to connect the moving parts

Finally, go to: `Services --> HAProxy --> Settings --> Rules & Checks --> Rules`

Here we add the rules that decide what to do with the traffic based on our map files (or conditions if necessary).

Forwarding HTTP to HTTPS

First, we must create a HTTPtoHTTPS_rule that will forward all HTTP traffic to HTTPS port, so it can go to our HTTPS_frontend.

1. `Name` is `HTTPtoHTTPS_rule`.
2. `Select conditions` this is set to the condition made earlier, the `NoSSL_condition`.
3. `Execute function` are built into HAProxy, the one we want is `http-request redirect`.
4. `HTTP Redirect` this is part of the paramateres for the function we chose earlier. Copy & Paste:

```
scheme https code 301
```

5. Save.

Mapping the map file to PUBLIC_SUBDOMAINS_rule

The PUBLIC_SUBDOMAINS_rule maps our subdomains to our backends using the map file we created in the previous step.

Continuing in `Rules`. Make a new rule for mapping domains to backends using a map file.

1. `Name` is `PUBLIC_SUBDOMAINS_rule`
2. `Execute function` must be set to `Map domains to backend pools using a map file`.
3. `Map file` must also be set, the map file made above was named `PUBLIC_SUBDOMAINS_mapfile`.
4. Save.
5. Apply.

> This is the meat and potatos of the tutorial. You will either get satiated or sick.

Adding the application service backend

First go to: `Services --> HAProxy --> Settings --> Real Servers`

This is where all of the servers on your network live. These are the “real servers” that exist on your network to which HAProxy can communicate with inorder to faciliate a reverse proxy. Note: unless you’re encrypting from the server to HAProxy with an internal CA, you dont need to verify SSL certificates.

The Virtual IP SSL_server

1. `Name` add `SSL_server` for our virtual IP address we set earlier.
2. `IP` will be the address of the virtual IP `127.1.2.3`
3. Uncheck `Verify SSL Certificate`
4. Save.
5. Apply.

Adding application service servers you’re hosting

1. `Name` add the name of your service for reference here, example: `NEXTCLOUD_server`
2. `IP` enter the IP address of the machine you want to reverse proxy.
3. `Port` enter the port number for the service(s) you want to proxy.
4. Uncheck `Verify SSL Certificate`
5. Save.
6. Apply.

Loadbalance SSL_backend pool

Next go to: `Services --> HAProxy --> Settings --> Virtual Services --> Backend Pools`

These are servers that lie in the backend pool. The backend pool cares for health monitoring and load distribution. If you had multiple servers sending out the same content, they’d reside in the same pool group.

> A Backend Pool must be configured, even if you only have one server.

Next, we will create the SSL_backend. This is the backend to which the SNI_frontend sends most of its traffic to.

Adding backend pools

SSL_backend

2. `Name` use `SSL_backend` for the first pool.
3. `Mode` is set to `TCP (Layer 4)` for the SSL_backend.
4. `Proxy Protocol` needs to be configured for `Version 2`.
5. `Servers` should be set to `SSL_server` made earlier, the one you made with the virtual IP `127.1.2.3`.
6. Save.

> Make sure that “SSL_backend” above is set to TCP mode, since the “SNI_frontend” is also running in TCP mode and you can’t mix HTTP mode and TCP mode with a frontend to backend.

Adding application service backends

Now we **create** the **backend** that belongs to an actual **service**.

- You will need one backend for each service.
- **Make SURE the backend is named the same as the one in your mapfile!**

1. `Name` should reference the service you’re adding, example: `NEXTCLOUD_backend`
2. `Mode` needs to be `HTTP (Layer 7) [default]`, this is different from the SSL_backend.
3. `Servers` set to the corresponding `Real Server` you made earlier.
4. Save.
5. Apply.

Note: If you have multiple servers serving the exact same content than you will want to add all servers into a single backend so HAProxy can actually balance the load between the servers.

Life without a map file

Setting an application’s condition to a backend pool… without using a map file… if you so desire

> This step is not required if you used the map file, this is an optional step!

If you didnt use a map file, or had a more particular configuration setup than “*starts with*”, feel free to set `rules` to point a `condition` to a `backend pool` that you made.

The `Condition` is generally made for the `condition type`. Again, as mentioned above, `Host starts with`.

The `Rule` states if the condition is met to execute a function. The function being, `Use specified Backend Pool`.

HAProxy Public Services

Next, go to: `Services --> HAProxy --> Settings --> Virtual Services --> Public Services`

> You will have to place the rules for all of your services that you don't want to get SSL offloaded in here.

At first we will create our SNI_frontend which will decide whether the traffic is going to be SSL offloaded or not.

Our default backend in this frontend will be the SSL_backend, that redirects all traffic to the virtual SSL_server which is actually the HTTPS_frontend.

Server Name Indicator frontend service

1. `Name` is `0_SNI_frontend`
2. `Description` should be `Listening on 0.0.0.0:443, 0.0.0.0:80` because you dont get to see the Listen Addresses for Public Services. It's a convience thing.
3. `Listen Addresses` need to be typed in, begin with `0.0.0.0:443` and then enter `0.0.0.0:80`
4. `Type` needs to be `TCP`
5. `Default Backend Pool` is for the `SSL_backend`
6. Save.
7. Apply.

HTTP redirect to HTTPS frontend service

Now we will create our HTTP_frontend.

Make sure to place the `HTTPtoHTTPS_rule` in this frontend!

This frontend is necessary in order to redirect HTTP traffic to HTTPS. But you could also use it to serve non SSL encrypted services on port 80.

1. Create a new `Public Service` and in the upper left hand corner of the create page, find `advanced mode` and turn it on (green).
2. `Name` is `1_HTTP_frontend`
3. `Listen Addresses` is `127.1.2.3:80` this combines our localhost address, `127.1.2.3` plus the HTTP port `:80`.
4. `Bind option pass-through` must be:

```
Plaintext
accept-proxy
```

5. Checkbox `Enable HTTP/2`
6. Also, checkbox `X-Forwarded-For header`
7. `Select Rules` needs to have `HTTPtoHTTPS_rule` set by clicking inside the square and selecting it.
8. Save.
9. Apply.

Now, the **big** show, create an “HTTPS_frontend”.

- This will be the primary frontend for traffic.
- It will be doing SSL offloading using the Let’s Encrypt certificate from the begining of this tutorial.
- You will place the “PUBLIC_SUBDOMAINS_rule” and any other rules for services that you want to get SSL offloaded in here.

1. Create a new `Public Service`
2. In the upper left hand corner of the edit page, find `advanced mode` and turn it on (green).
3. `Name` is `1_HTTPS_frontend`
4. `Listen Addresses` is `127.1.2.3:443` this combines our localhost address, `127.1.2.3` plus the HTTPS port `:443`.
5. `Bind option pass-through` must be:

```
accept-proxy
```

6. Checkbox `Enable SSL offloading`
7. A new settings area, ‘SSL Offloading,’ will appear, and you will need to select your Let’s Encrypt certificate under the `Certificates` section.
8. `SSL option pass-through` should be set to:

```
curves secp384r1
```

9. `Enable Advanced settings` needs to be checked.
    - This opens a NEW section for Cipers

[Current Ciphers and Cipher Suites for a 100% A+ SSLLabs rating]

[Last updated/verified on 20230223 using Mozilla SSL Configuration Generator.]

> All ciphers with a strength of 128 bit or below have been removed in order to get a 100% A+ rating at SSL Labs.

```
Cipher List
ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-
POLY1305:DHE-RSA-AES256-GCM-SHA384

Cipher Suites
TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256
```

1. `Cipher List` should read from the code block above `ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-AES256-GCM-SHA384`
2. `Cipher Suites` should also read from the code block above `TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256`

4. `HSTS preload` also needs checked.
5. `HSTS max-age` can be increased to `63072000`
6. `Bind options` need to have `prefer-client-ciphers` removed, and add `no-sslv3`, `no-tlsv10`, `no-tlsv11`, `no-tls-tickets`
7. Checkbox `Enable HTTP/2`
8. Also, checkbox `X-Forwarded-For header`
9. `Select Rules` needs to have `PUBLIC_SUBDOMAINS_rule` set by clicking inside the square and selecting it.
10. Save.
11. Apply.

Update Map file? Restart Service!

Anytime you want to add a new service and subdomain here are the steps:

1. Create a `Real Server` that points to the IP address and PORT of the service that is running (example: I'm running Proxmox and want to verified SSL that service. I'd find the IP address of the machine, and the port number of the service you're trying to connect to. In Proxmox's case this is: [IP - 192.168.1.12 with a PORT 8006] and, as Proxmox is encrypted by default, we must check the `SSL` checkbox to tell to HAProxy to expect an SSL connection. MAKE SURE to UNCHECK `Verify SSL Certificate`.)
    - That's it! There are a lot of options, but all the `Real Server` needs is IP and PORT and if that port is SSL or not.
2. Creating a new `Backend Server`. Enter YOURSERVICENAMEHERE_backend for the name of this backend server. Then, select the name of the Real Server created above in the `Servers` section.
    - This is where your pool would live if you had multiple servers doing the same thing.
3. Modifing the `Rules` file. This is why it's important to be consistant with names, we now must enter the subdomain to match with a backend. Naming our backends consistantly helps ensure when we need to add/edit the map file, it's seemless.
    - The map file matches the entered subdomain with the backend service.
4. **Restart HAProxy.**
    - That's right nothing will work unless HAProxy loads the new map file.

Life without a map file: part 2

Setting a certificate for applications' backend pool... without using a map file... if you so desire

If you didnt use a map file **and** are **not** going to point your traffic to a Virtual IP, then these instructions should get you up and running.

1. `Name` set it to anyting you like, example: `Public_Faceing_Pool`.
2. `Listen Address` should be your Public IP Address with port 443 on the end, example: `123.45.67.89:443`.
3. `Type` is `HTTP/HTTPS (SSL offloading) [default]`.
4. `Default Backend Pool` can be set to any backend pool you made earlier – for testing of course, as ALL subdomains will connect to this one backend that then finds the server.

6. `Certificates` if you click in the box, you should see the certificate name you used above.
7. `Default certificate` this is a drop down, choose.
8. `Select Rules` use the rule that connects the condition to the backend pool.
9. Save.

Test your new certificate

Access from external networks should now already be working.

Just try to access your URL “nextcloud.test.house.lan” from any device that is not connected to your local network, a good test device is your smartphone on cellular data.

You can now test your SSL settings at: https://www.ssllabs.com/ssltest/index.html

---

Access to your services, directly from internal networks

Choosing a DNS solution

If you try to access your URL “nextcloud.test.house.lan” from a device in your internal network, it should fail. Not just because that’s not a real TLD, but because your search domain is in your local network.

There are two ways of fixing this.

- With (Option A) being the better of the two. * Option A - Split DNS
- With (Option B) you lose the ability to track originating source IP in HAProxy when going through NAT.
- Option B - NAT Reflection

Option A - Split DNS (DNS Overrides)

Option A presumes you can create DNS entries on your local network’s DNS.

`Unbound DNS` will easily set up DNS overrides.

To set this up, go to: `Services --> Unbound DNS --> Overrides`

- Here you will need to create “Host Overrides” for each of your services.
- Yes, **every subdomain** that you want to seperate from the upstream public DNS.

The IP address can be any LAN (or VLAN) interface IP of your OPNsense.

I am using the LAN IP for OPNsense, 192.168.1.254 on which the `0_SNI_frontend` is also listening on.

1. `Host` set the name to `*`
2. `Domain` should be the subdomain you registered the Let’s Encrypt certificate for, example: “test.house.lan”.
3. `Type` is an `A (IPv4 address)`

5. Save.

Option B - NAT Reflection

This option uses firewall rules to check NAT first.

Please note that “NAT Reflection” is only applicable when port forwarding.

Open: `Firewall --> NAT --> Port Forward`

Create a new rule with the following:

1. `Interface` is set to `WAN`
2. `Destination` needs to be `WAN address`
3. `Destination port range` should be set to the alias, `HAProxy_Ports`
4. `Redirect target IP` should be one of the Virtual IPs we set earlier that the `0_SNI_frontend` is listening on.
5. `NAT reflection` must changed to be `Enable` for this to work.
6. Save.

Advanced Configuration: local-access-only subdomains

Imagine you have a service that you would like to access / protect using your brand new reverse proxy without making it available on the internet?

Well, HAProxy has got you covered!

Map file for local only subdomains

Going back to the map files section: `Services --> HAProxy --> Settings --> Advanced --> Map Files`

1. Clone the current, `PUBLIC_SUBDOMAINS_mapfile`, rename it to `LOCAL_SUBDOMAINS_mapfile`.
2. Add all of the subdomains you want to be local-access-only along with their corresponding backends.

> Keep in mind that the contents of your “PUBLIC_SUBDOMAINS_mapfile” must also be in the “LOCAL_SUBDOMAINS_mapfile”.

HAProxy is processing the rules in the frontends based on the order they appear.

So if you place your `PUBLIC_SUBDOMAINS_rule` before your `LOCAL_SUBDOMAINS_rule` in the frontend configuration, **you won’t get access to your local-access-only subdomains**.

> Vice versa this will also happen and you will no longer have access to your public subdomains.

To **avoid this** you have to also put the content of your `PUBLIC_SUBDOMAINS_mapfile` in the `LOCAL_SUBDOMAINS_mapfile` and place their **rules in the correct order**.

```Plaintext
# local access only subdomains
localstuff MYSTUFF_backend

# public access subdomains
nextcloud NEXTCLOUD_backend
```

Create a local address condition

Conditions are in HAProxy: `Services --> HAProxy --> Settings --> Rules & Checks --> Conditions`

Now you need to create a new condition that detects if the source of the request is a local IP.

1. `Name` is set set to `LOCAL_SUBDOMAINS_SUBNETS_condition`
2. `Condition type` should be `Source IP matches specified IP`
3. The `Source IP` should be as small and specific to your network as possible, or just use the entire RFC1918 IP range:

```Plaintext
10.0.0.0/8 172.16.0.0/12 192.168.0.0/16
```

You can use the predefined “Source IP is local” condition instead if needed.

Use DNS addresses for the condition

If you *really* need to, you can use DNS entries to match IPs that should be able to access local services.

But the resolving is only done once during the start / restart of HAProxy.

Adding the LOCAL_SUBDOMAINS_rule

Move over to rules: `Services --> HAProxy --> Settings --> Rules & Checks --> Rules`

1. Clone the current, `PUBLIC_SUBDOMAINS_rule`, rename it to `LOCAL_SUBDOMAINS_rule`
2. `Name` is `LOCAL_SUBDOMAINS_rule`
3. `Select Conditions` and check/select the `LOCAL_SUBDOMAINS_SUBNETS_condition` you made earlier.
   - If you are using more than one set of SUBDOMAIN_conditions, you will need to change the `Logical operator for conditions` to `OR`.
4. `Execute function` needs to be set to `Map domains to backend pools using a map file`
5. `Map file` should be set to the name of our cloned and recreated mapfile, `LOCAL_SUBDOMAINS_mapfile`
6. Save.

Ordering your HTTPS frontend rules

Final step: `Services --> HAProxy --> Settings --> Virtual Services --> Public Services`

1. Edit your `1_HTTPS_frontend`

3. Make sure you have your `LOCAL_SUBDOMAINS_rule` first.
    - This needs to be in the upper left corner, as in, first.

> ! Make sure your `LOCAL_SUBDOMAINS_rule` comes before your `PUBLIC_SUBDOMAINS_rule` entry in the rules for your HTTPS frontend public service.

Hide your certificate association with your WAN address

You might have noticed that if you can access your OPNsense using your public WAN IP (https://YOUR_PUBLIC_IP/) the connection will be secured, but upon further inspection you will see that your Let's Encrypt certificate tied to your domain is beeing used.

While this is not a major security problem it still presents at least some privacy issues.

To fix this we can present a dummy certificate to everyone accessing your reverse proxy directly using your public WAN IP (https://YOUR_PUBLIC_IP/).

Create a placeholder certificate

Create a certificate authority

Create a new CA: `System --> Trust --> Authorities`

Here we have to create a placeholder certificate authority in order to create the placeholder certificate in the next step.

1. `Name` will be `Invalid_SNI`
2. `Method` is set to `Create an internal Certificate Authority`
3. `Key Type` should be `Elliptic Curve`
4. `Curve` needs to be `secp521r1`
5. `Digest Algorithm` is `SHA512`
6. `Lifetime` can be `10950`
7. Fill in the rest of the fields with: `Invalid_SNI`
8. Save.

Create a certificate

Now, the certificate: `System --> Trust --> Certificates`

Create the placeholder certificate.

1. `Method` is `Create an internal Certificate`
2. `Descriptive Name` is `Invalid_SNI`
3. `Certificate authority` is also `Invalid_SNI`

5. `Key Type` is `Elliptic Curve`
6. `Curve` is `secp521r1`
7. `Digest Algorithm` is `SHA512`
8. `Lifetime` can be `10950`
9. `Private key location` should be `Save on this firewall`
10. Fill in the rest of the fields with: `Invalid_SNI`.
11. Save.

Connecting your frontend public service to a default placeholder certificate

Finally, get to: `Services --> HAProxy --> Settings --> Virtual Services --> Public Services`

The last thing left to do is to configure the placeholder certificate as “Default certificate” in your `1_HTTPS_frontend`.

1. Open your `1_HTTPS_frontend` in your `Public Services`
2. Scroll to the `Default certificate` section.
3. Choose our newly minted placeholder certificate, `Invalid_SNI`

You should now no longer get presented with your trusted Let’s Encrypt certificate when accessing “https://YOUR_PUBLIC_IP”, but instead with the “Invalid_SNI” certificate. Thus masking your IP address connected to your domains.

Sources:

- TheMaw Tech for the basic pfsense style HAProxy Setup
- https://www.youtube.com/watch?v=uACQrhtsgFk
- TheHellSite forum post on OPNsense’s forums
- https://forum.opnsense.org/index.php?topic=23339.0
- OPNsense’s documentation
- https://docs.huihoo.com/m0n0wall/opnsense/manual/how-tos/haproxy.html

Networking, Router

opnsense haproxy acme reverse-proxy split-dns free-certificates

Further Reading

Visual Guide to OPNsense multi-site with HAProxy, Unbound

Screenshot tutorial to use OPNsense for a Reverse Proxy using multiple domains with…

OPNsense HAProxy Proxy Protocol to Traefik with original IP

HAProxy proxy protocol section Different backends based on URL. Domain specific…

Unbound Views in OPNsense to Resolve Domains by VLAN/Subnet

Configure Unbound DNS in OPNsense for Subnet Based Domain Resolution Running…

OLDER
LUKS Encrypt Linux Laptop

NEWER
Rename Nodes on a Proxmox Cluster