

Setting up Self-Signed Multiple FQDN Certificates for Local Services in Traefik



In this article, we will walk through creating a self-signed certificate for multiple local services (e.g., Portainer and Pi-hole) using OpenSSL. We'll also configure Traefik to use this certificate in Docker Swarm. Additionally, we will explain how SSL certificates work, the role of key components like the private key, public key, and Certification Authority (CA). We'll use the provided image for understanding these concepts.

Prerequisites

- You already have [Traefik](#), [Portainer](#), [Pi-hole](#), and [Docker Swarm](#) set up.
- You have a local DNS setup using Pi-hole to resolve local domain names such as `portainer.local` and `pihole.local`.

Whats FQDN?

FQDN (Fully Qualified Domain Name) is the complete domain name of a specific host within the internet or a local network. It includes both the hostname and the domain name, ensuring the address is globally unique. An FQDN typically follows this format: `hostname.domain.tld` (e.g., `www.example.com`). For local networks, it can be something like `portainer.local` or `pihole.local`. The FQDN provides a precise location for a resource in the DNS hierarchy, making it essential for properly identifying services across networks.

Step 1: Create a Multiple FQDN Certificate with OpenSSL

1. **Generate a private key** for the certificate:

```
openssl genrsa -out local.key 4096
```

2. **Create a Certificate Signing Request (CSR) for multiple FQDNs.** First, create a configuration file `san.cnf`:

```
touch /mnt/glustermount/data/certs/san.cnf
```

```
[req] # Request options
default_bits      = 4096 # Size of the encryption key
prompt           = no   # No prompts, all values are provided in the config file
default_md       = sha256 # Use SHA256 for the certificate
distinguished_name = dn  # Use the 'dn' section for distinguished names
req_extensions    = req_ext # Use 'req_ext' for additional extensions like SAN (Subject Alternative Name)

[dn] # Distinguished Name section
CN = portainer.local # Common Name (CN) for the certificate (primary domain)

[req_ext] # Extensions for the certificate request
subjectAltName = @alt_names # Use alternative names (SAN)

[alt_names] # Alternative domain names
DNS.1 = portainer.local # First DNS name (alternative domain)
DNS.2 = pi-hole.local   # Second DNS name (alternative domain)
```

3. Generate the CSR using the configuration file:

```
openssl req -new -key local.key -out local.csr -config san.cnf
```

4. Generate a self-signed certificate for 1 year (365 days):

```
openssl x509 -req -in local.csr -signkey local.key -out local.crt -days 365 -extfile san.cnf -extensions req_ext
```

5. Move the certificate and key to a shared directory accessible by Docker Swarm:
(If you need help to understand how the Nodes of the Docker Swarm Cluster are sharing the synced Files - [Check this Article](#))

```
mkdir -p /mnt/glustermount/data/certs
mv local.crt local.key /mnt/glustermount/data/certs/
```

Step 2: Understanding the `san.cnf` File

The `san.cnf` file helps OpenSSL create a certificate with multiple domain names (FQDNs). Here's a breakdown of the file:

- **[req]:** Specifies the general options for generating the certificate request, such as key size, hashing algorithm (SHA256), and the distinguished name section.
- **[dn]:** Defines the common name (CN), which in this case is the primary domain (`portainer.local`).
- **[req_ext]:** Specifies the Subject Alternative Names (SANs), which allow the certificate to be valid for additional domain names (e.g., `pihole.local`).
- **[alt_names]:** Lists the additional domain names (`DNS.1`, `DNS.2`, etc.) that will be included in the certificate.

This setup creates a certificate that can be used for both `portainer.local` and `pihole.local`, ensuring secure access over HTTPS. You can add any other Local DNS Entry to the List. Just make sure to add the Entry to your Pihole under the "Local DNS - DNS Records" Section.

Step 3: Add the Certificate to Traefik's Static Configuration (TOML)

Edit your `traefik.toml` file to include the certificate you generated:

```
[entryPoints]
[entryPoints.websecure]
  address = ":443"

[tls]
[[tls.certificates]]
  certFile = "/mnt/glustermount/data/certs/local.crt"
  keyFile = "/mnt/glustermount/data/certs/local.key"
  stores = ["default"]
```

```
[tls.stores]
[tls.stores.default]
[tls.stores.default.defaultCertificate]
  certFile = "/mnt/glustermount/data/certs/local.crt"
  keyFile  = "/mnt/glustermount/data/certs/local.key"
```

This configuration tells Traefik to use the multiple FQDN certificate (`local.crt`) for requests matching `portainer.local` and `pihole.local` .

Step 4: Assign the Self-Signed Certificate to Specific Services

Now, configure the dynamic behavior of Traefik using the `dynamic.toml` file:

```
[http]
[http.routers]
[http.routers.portainer-secure]
  rule = "Host(`portainer.local`)"
  service = "portainer"
  entryPoints = ["websecure"]
  tls = { certResolver = "self-signed" }

[http.routers.pihole-secure]
  rule = "Host(`pihole.local`)"
  service = "pihole"
  entryPoints = ["websecure"]
  tls = { certResolver = "self-signed" }

[http.services]
[http.services.portainer.loadBalancer]
  [[http.services.portainer.loadBalancer.servers]]
    url = "http://portainer:9443"

[http.services.pihole.loadBalancer]
  [[http.services.pihole.loadBalancer.servers]]
    url = "http://pihole:888"
```

Step 5: Adjust the Pi-hole Docker Compose Configuration

Here's the Pi-hole `docker-compose.yml` adjusted to match the certificate and Traefik settings:

```
version: '3'

services:
  pihole:
    networks:
      - management_net # For management via Traefik
    image: pihole/pihole:latest
    ports:
      - "53:53/tcp"
      - "53:53/udp"
      - "888:80"
    environment:
      TZ: 'Europe/Zurich'
      WEBPASSWORD: '${PIHOLE_PASSWORD}'
    volumes:
      - '/mnt/glustermount/data/pihole_data/etc:/etc/pihole'
      - '/mnt/glustermount/data/pihole_data/dns:/etc/dnsmasq.d'
    restart: unless-stopped
    deploy:
      mode: replicated
      replicas: 1
      placement:
        constraints: [node.platform.os == linux]
      labels:
        - 'traefik.enable=true'
        - "traefik.http.routers.pihole-secure.rule=Host(`pihole.local`)"
        - "traefik.http.routers.pihole-secure.entrypoints=websecure"
        - "traefik.http.routers.pihole-secure.tls=true"
        - "traefik.http.services.pihole.loadbalancer.server.port=80"

networks:
  management_net:
    external: true
```

Step 6: Deploy the Updated Stack

Run the following command to apply the updated stack configuration:
(You can also use Portainer running the Stack)

```
docker stack deploy -c docker-compose.yml your_stack_name
```

Step 7: Test the Setup

1. **Add the self-signed certificate to your trusted sources** on your machine. This can be done by importing the `.crt` file into your browser or system's trusted certificates store.
2. **Verify the secure connections:**
 - Access `https://portainer.local:9443`
 - Access `https://pihole.local:888`

Both should now use your self-signed certificate with proper encryption.

Revision #5

Created 7 October 2024 15:47:20 by aeoneros

Updated 5 January 2025 11:17:02 by aeoneros