

BTS SIO

BLOC 3 OPTION SISR - TP

© GH 01/2026

TESTS DE COMPREHENSION TRAEFIK

TP 1 – SOUS-DOMAIN SANS ROUTEUR/SERVICE TRAEFIK ET SNI STRICT

Contexte

Vous disposez d'un reverse proxy **Traefik** exposé sur Internet. Plusieurs sous-domaines ont été enregistrés chez un hébergeur et pointent vers votre IP publique (WAN). Ces sous-domaines vous ont été communiqués précédemment. Traefik décide quel service publier grâce à des « **routers** ».

Objectifs

1. Déployer Apache dans Docker (**mappage de port :8083**).
2. Prouver qu'un sous-domaine ne suffit pas sans router Traefik.
3. Ajouter un router Traefik + ACME pour obtenir un certificat Let's Encrypt et publier le site.
4. Valider avec des tests factuels.

Pré-requis

- Un sous-domaine public : webtest.teamX.sio-ndlp.fr (record A vers IP WAN).

1^{ère} étape : déployer le site de test

Créez votre arborescence sur votre machine Docker (remplacez le X par le numéro de votre équipe) :

```
sudo mkdir -p /opt/webtest-teamX/site  
cd /opt/webtest-teamX/site
```

Créez une page d'accueil :

```
sudo nano index.html
```

Contenu à copier (modifiez les valeurs en **rouge**) :

```
<!doctype html>  
<html lang="fr">  
<head>  
  <meta charset="utf-8">  
  <title>SITE DE TEST</title>  
</head>  
<body>  
  <h1>SITE DE TEST</h1>  
  <p>Domaine : webtest.teamX.sio-ndlp.fr</p>  
</body>  
</html>
```

Créez le docker compose dans le dossier /opt/webtest-teamX :

Contenu à copier (modifiez la valeur en rouge) :

```
services:
  webtest:
    image: httpd:2.4-alpine
    container_name: webtest-teamX
    restart: unless-stopped
    ports:
      - "8083:80"
    volumes:
      - ./site/index.html:/usr/local/apache2/htdocs/index.html:ro
```

Lancez la création du conteneur avec « **docker compose up -d** ».

Vérifiez que le conteneur est bien en mode « Up » avec la commande « **docker ps -a** » :

					NAMES
bdb125b61118	httpd:2.4-alpine	"httpd-foreground"	About a minute ago	Up About a minute	0.0.0.0:8083
3->80/tcp, [::]:8083->80/tcp				webtest-prof	

2^{ème} étape : tester le backend depuis la machine Docker

Vérifiez le bon fonctionnement du « backend » depuis la machine Docker avec la commande suivante :

curl -I http://127.0.0.1:8083

Retour attendu :

```
docker@DEBIAN-DOCKER-PROF:/opt/webtest-prof$ curl -I http://127.0.0.1:8083
HTTP/1.1 200 OK
Date: Wed, 24 Dec 2025 10:03:47 GMT
Server: Apache/2.4.66 (Unix)
Last-Modified: Wed, 24 Dec 2025 09:55:48 GMT
ETag: "c2-646afaa2c3f77"
Accept-Ranges: bytes
Content-Length: 194
Content-Type: text/html
```

3^{ème} étape : vérification de la résolution DNS

Vérifiez que le DNS est bien résolu depuis votre poste Windows avec la commande suivante :

nslookup webtest.teamX.sio-ndlp.fr

Retour attendu :

```
PS C:\Users\pc> nslookup webtest.prof.sio-ndlp.fr
Serveur : dns9.quad9.net
Address: 9.9.9.9

Réponse ne faisant pas autorité :
Nom : webtest.prof.sio-ndlp.fr
Address: 212.83.149.102
```

4^{ème} étape : test de connexion

Depuis un navigateur, si vous saisissez l'URL <https://webtest-teamX.sio-ndlp.fr> vous n'obtiendrez pas d'accès puisque le site n'est pas publié derrière Traefik (pas de certificat Let's Encrypt) :



5^{ème} étape : création du « routeur/service » Traefik et tests

Créez le fichier de configuration dynamique nécessaire dans Traefik. Une fois le fichier créé, testez l'accès au site de test et vérifiez l'obtention du certificat Let's Encrypt :

SITE DE TEST

Domaine : webtest.prof.sio-ndlp.fr

Émis pour

Nom commun (CN)

webtest.prof.sio-ndlp.fr

Émis par

Nom commun (CN)

R13

Organisation (O)

Let's Encrypt

On peut voir que le certificat est présent dans le fichier « acme.json » de Traefik avec la commande suivante (modifier **.teamX**):

sudo grep -n "webtest\..teamX\.sio-ndlp\.fr" /var/lib/traefik/acme.json

```
traefik@PVESIO2-DEBIAN-TRAEFIK-PROF:/etc/traefik/dynamic$ sudo grep -n "webtest\..prof\.sio-ndlp\.fr" /var/lib/traefik/acme.json
49:      "main": "webtest.prof.sio-ndlp.fr"
```

Contexte

Traefik peut ajouter des **en-têtes HTTP de sécurité** aux réponses grâce à un middleware (ici SecureHeaders). L'objectif est de **mesurer** la différence entre un site "sans headers" et "avec headers", puis d'expliquer l'intérêt des principaux en-têtes.

Objectifs

- Vérifier la présence des headers avant/après activation du middleware SecureHeaders.
- Comprendre le rôle de HSTS, CSP, Permissions-Policy, nosniff, X-Frame-Options, Referrer-Policy.
- Relier le résultat aux fichiers de configuration Traefik.

Pré-requis

- Votre site <https://webtest.teamX.sio-ndlp.fr> fonctionne (TP1 terminé).
- Accès à un terminal Windows PowerShell.

1^{ère} étape : vérification des « headers » utilisés

Depuis Powershell, saisissez la commande suivante (modifiez les valeurs en **rouge**) :

```
(Invoke-WebRequest -UseBasicParsing -Uri "https://webtest.teamX.sio-ndlp.fr" -Method Head).Headers
```

Retour attendu :

```
PS C:\Users\pc> (Invoke-WebRequest -UseBasicParsing -Uri "https://webtest.prof.sio-ndlp.fr" -Method Head).Headers

Key           Value
---           -
Alt-Svc       h3=":443"; ma=2592000
Accept-Ranges bytes
Content-Length 194
Content-Type   text/html
Date          Wed, 24 Dec 2025 11:02:22 GMT
ETag          "c2-646afaa2c3f77"
Last-Modified Wed, 24 Dec 2025 09:55:48 GMT
Server        Apache/2.4.66 (Unix)
```

Ici on voit le « Server: » Apache/2.4.66 (Unix) et « Alt-Svc: » h3=":443" : ce qui confirme que Traefik sert bien en frontal (HTTP/3 annoncé) et que l'on a atteint le backend. Il manque juste les headers de sécurité car notre fichier de configuration dynamique (routeur/service) n'a pas de « headers » pour le moment. On ajoute les « headers » à ce fichier (voir lignes en **rouge** ci-dessous) :

http:

 routers:

 webtest-prof:

 entryPoints: ["websecure"]

 rule: "Host(`webtest.prof.sio-ndlp.fr`)"

 service: webtest-prof-svc

middlewares:

- SecureHeaders

 tls:

 certResolver: le

 options: default

 services:

 webtest-prof-svc:

loadBalancer:
servers:
- url: <http://192.168.100.110:8083>

Si on saisit à nouveau la commande, on va maintenant voir que les « headers » ont bien été appliqués :

(Invoke-WebRequest -UseBasicParsing -Uri "https://webtest.teamX.sio-ndlp.fr" -Method Head).Headers

Retour attendu :

```
PS C:\Users\pc> (Invoke-WebRequest -UseBasicParsing -Uri "https://webtest.prof.sio-ndlp.fr" -Method Head).Headers
```

Key	Value
Alt-Svc	h3=":443"; ma=2592000
Content-Security-Policy	default-src 'self'; script-src 'self' 'unsafe-inline' 'unsafe-eval'; style-src 'self' 'unsafe-inline'; im...
Permissions-Policy	accelerometer=(), camera=(), geolocation=(), gyroscope=(), magnetometer=(), microphone=(), payment=(), us...
Referrer-Policy	same-origin
Strict-Transport-Security	max-age=63072000; includeSubDomains; preload
X-Content-Type-Options	nosniff
X-Frame-Options	SAMEORIGIN
Accept-Ranges	bytes
Content-Length	194
Content-Type	text/html
Date	Wed, 24 Dec 2025 11:13:13 GMT
ETag	"c2-646afaa2c3f77"
Last-Modified	Wed, 24 Dec 2025 09:55:48 GMT
Server	Apache/2.4.66 (Unix)

Appliquez les « headers » sur le site 1 préalablement créé dans le 1^{er} TP et vérifiez l'application des « headers » pour ce site :

Retour attendu (site 1) :

```
PS C:\Users\pc> (Invoke-WebRequest -UseBasicParsing -Uri "https://web1.prof.sio-ndlp.fr" -Method Head).Headers
```

Key	Value
Alt-Svc	h3=":443"; ma=2592000
Content-Security-Policy	default-src 'self'; script-src 'self' 'unsafe-inline' 'unsafe-eval'; style-src 'self' 'unsafe-inline'; im...
Permissions-Policy	accelerometer=(), camera=(), geolocation=(), gyroscope=(), magnetometer=(), microphone=(), payment=(), us...
Referrer-Policy	same-origin
Strict-Transport-Security	max-age=63072000; includeSubDomains; preload
X-Content-Type-Options	nosniff
X-Frame-Options	SAMEORIGIN
Accept-Ranges	bytes
Content-Length	233
Content-Type	text/html
Date	Wed, 24 Dec 2025 11:19:47 GMT
ETag	"e9-646a1c22f4f71"
Last-Modified	Tue, 23 Dec 2025 17:20:21 GMT
Server	Apache/2.4.66 (Unix)

Retour attendu (site 2 sans « headers ») :

```
PS C:\Users\pc> (Invoke-WebRequest -UseBasicParsing -Uri "https://web2.prof.sio-ndlp.fr" -Method Head).Headers
```

Key	Value
Alt-Svc	h3=":443"; ma=2592000
Accept-Ranges	bytes
Content-Length	233
Content-Type	text/html
Date	Wed, 24 Dec 2025 11:21:42 GMT
ETag	"e9-646a1c4dc30ba"
Last-Modified	Tue, 23 Dec 2025 17:21:06 GMT
Server	Apache/2.4.66 (Unix)

Retour attendu (site 2 avec « headers ») :

```
PS C:\Users\pc> (Invoke-WebRequest -UseBasicParsing -Uri "https://web2.prof.sio-ndlp.fr" -Method Head).Headers
```

Key	Value
Alt-Svc	h3=":443"; ma=2592000
Content-Security-Policy	default-src 'self'; script-src 'self' 'unsafe-inline' 'unsafe-eval'; style-src 'self' 'unsafe-inline'; im...
Permissions-Policy	accelerometer=(), camera=(), geolocation=(), gyroscope=(), magnetometer=(), microphone=(), payment=(), us...
Referrer-Policy	same-origin
Strict-Transport-Security	max-age=63072000; includeSubDomains; preload
X-Content-Type-Options	nosniff
X-Frame-Options	SAMEORIGIN
Accept-Ranges	bytes
Content-Length	233
Content-Type	text/html
Date	Wed, 24 Dec 2025 11:22:42 GMT
ETag	"e9-646a1c4dc30ba"
Last-Modified	Tue, 23 Dec 2025 17:21:06 GMT
Server	Apache/2.4.66 (Unix)

Avec Traefik, il existe un middleware nommé « **Compression** ». Ce middleware sert à **compresser la réponse HTTP** avant de l'envoyer au client, afin de :

- **réduire la taille** des pages et des ressources (HTML, CSS, JS, JSON...),
- **accélérer le chargement** côté client (moins de données à transférer),
- **diminuer la bande passante** consommée sur le lien WAN.

Comment ça fonctionne (mécanisme) :

Le navigateur envoie une requête avec un « header » du type : « Accept-Encoding: » gzip, br, deflate (selon le navigateur).

Traefik reçoit la réponse du backend (Apache, etc.).

Traefik compresse la réponse (si le type de contenu est éligible) et renvoie avec un contenu plus petit (compression).

C'est un bon exemple de middleware «**transformation de flux**» (contrairement à SecureHeaders qui ne modifie que les **headers**). Ça illustre que Traefik n'est pas qu'un routeur : il peut optimiser et normaliser le trafic. De manière générale, Traefik compresse :

- text/html, text/css, application/javascript
- application/json, text/plain, application/xml

A retenir :

- La compression dépend du client (**Accept-Encoding**).
- Elle améliore la performance pour les contenus texte (HTML/CSS/JS/JSON).
- On exclut certains contenus (SSE) pour éviter des effets indésirables.

C'est une optimisation centralisée au niveau du reverse proxy : utile quand il y a beaucoup de services.

Pour utiliser le middleware « compression », nous devons créer le fichier « yml » et ajouter le middleware au routeur/service concerné.

1^{ère} étape : création du middleware « compression »

- Sur votre machine Traefik, créez un nouveau fichier « mw-compression.yml » dans le dossier /etc/traefik/dynamic avec la commande suivante :

sudo nano /etc/traefik/dynamic/mw-compression.yml

Contenu à copier (en respectant l'indentation) :

```
http:
  middlewares:
    Compression:
      compress:
        excludedContentTypes:
          - text/event-stream
```

Remarque : « excludedContentTypes: text/event-stream » évite de casser/retarder les flux SSE.

2^{ème} étape : ajout du middleware « Compression » dans le routeur/service du site « webtest »

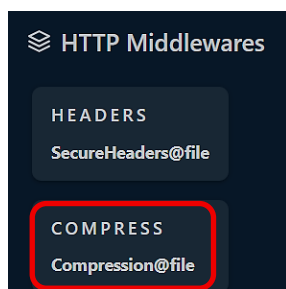
Ici, nous allons ajouter le middleware « Compression » dans le routeur/service du site webtest précédemment créé. Pour cela, éditez le fichier routeur/service de votre site de test et ajoutez ceci :

```
http:
  routers:
    webtest-prof:
      entryPoints: ["websecure"]
      rule: "Host(`webtest.prof.sio-ndlp.fr`)"
      service: webtest-prof-svc
      middlewares:
        - SecureHeaders
        - Compression
      tls:
        certResolver: le
        options: default

  services:
    webtest-prof-svc:
      loadBalancer:
        servers:
          - url: "http://192.168.100.110:8083"
```

On peut vérifier que le middleware compression a bien été pris en compte par Traefik depuis le tableau de bord :

- Cliquez sur « **HTTP Middlewares** »
- Vérifiez que le middleware « Compression » a bien été pris en compte (il apparaît sous la forme « COMPRESS – Compression@file ») :



Test de compression depuis le site web de test avec la commande suivante :

```
curl.exe -s -o NUL -H "Accept-Encoding: gzip" -w "gzip_size=%{size_download}\n" https://webtest.teamX.sio-ndlp.fr
```

```
PS C:\Users\pc> curl.exe -s -o NUL -H "Accept-Encoding: gzip" -w "gzip_size=%{size_download}\n" https://webtest.prof.sio-ndlp.fr
gzip_size=194
```

Test sur le site 1 ou le site 2 sur lequel il n'y a pas de compression :

```
curl.exe -s -o NUL -H "Accept-Encoding: gzip" -w "gzip_size=%{size_download}\n" https://web1.teamX.sio-ndlp.fr
```

```
PS C:\Users\pc> curl.exe -s -o NUL -H "Accept-Encoding: gzip" -w "gzip_size=%{size_download}\n" https://web1.prof.sio-ndlp.fr
gzip_size=233
```

La différence, ici, est peu significative car les sites sont quasi identiques mais on note quand même une légère différence.