PFSENSE

Adrien BLAIZE

SOMMAIRE

PREMIÈRE PARTIE

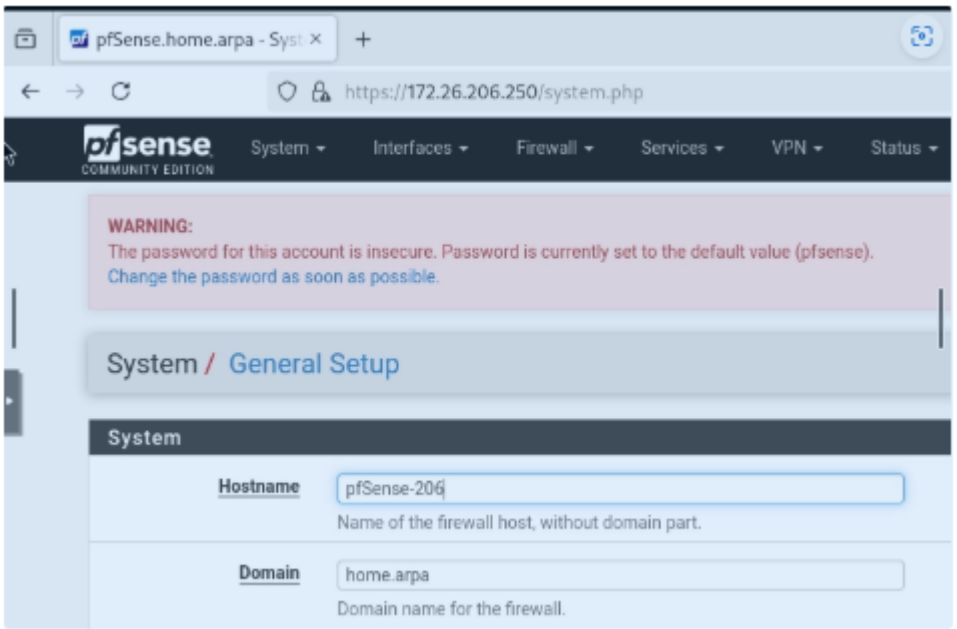
1 - Configuration des serveurs

```
saving to /etc/xrdp/rsakeys.ini

Created symlink '/etc/systemd/system/multi-user.target.wants/xrdp-sesman.service' → '/usr/lib/systemd/system/xrdp-sesman.service'.
Created symlink '/etc/systemd/system/multi-user.target.wants/xrdp.service' → '/usr/lib/systemd/system/xrdp.service'.
Traitement des actions différées (« triggers ») pour man-db (2.13.1-1) ...
Traitement des actions différées (« triggers ») pour libc-bin (2.41-12) ...
root@ablai-Apache:~# ^C
root@ablai-Apache:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: ens18: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether bc:24:11:1d:1b:4f brd ff:ff:ff:ff:ff:ff
    altname enp0s18
    altname enxbc24111d1b4f
    inet 172.26.206.80/24 brd 172.26.206.255 scope global ens18
       valid_lft forever preferred_lft forever
    inet6 fe80::be24:11ff:fe1d:1b4f/64 scope link proto kernel_ll
       valid_lft forever preferred_lft forever
root@ablai-Apache:~#
```

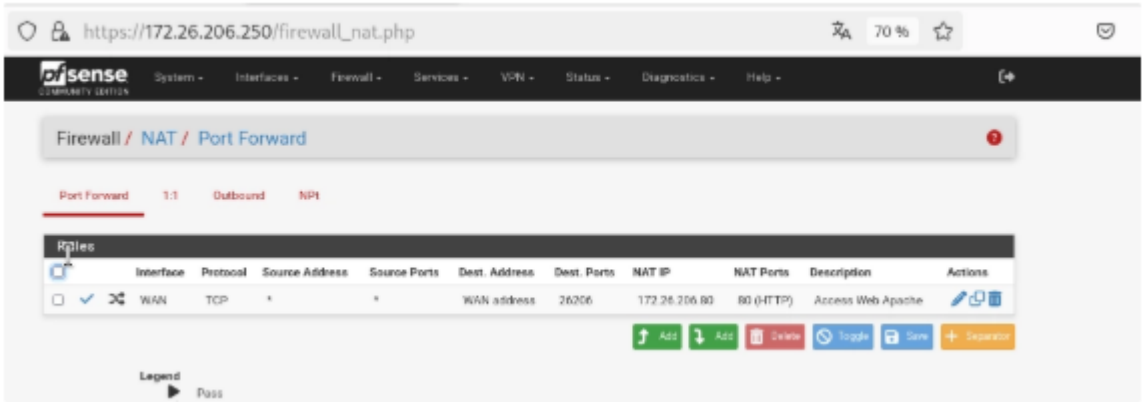
Configuration de l'interface réseau du serveur Apache (enp0s18).

Network Device (net0)	virtio=BC:24:11:14:04:0E,bridge=vlan992,firewall=1
Network Device (net1)	virtio=BC:24:11:AA:30:61,bridge=vlan206,firewall=1

Mise en place de deux cartes réseaux virtuels sur le serveur PFSENSE.



Ajout de l’hostname et du domaine dans System/General Setup.



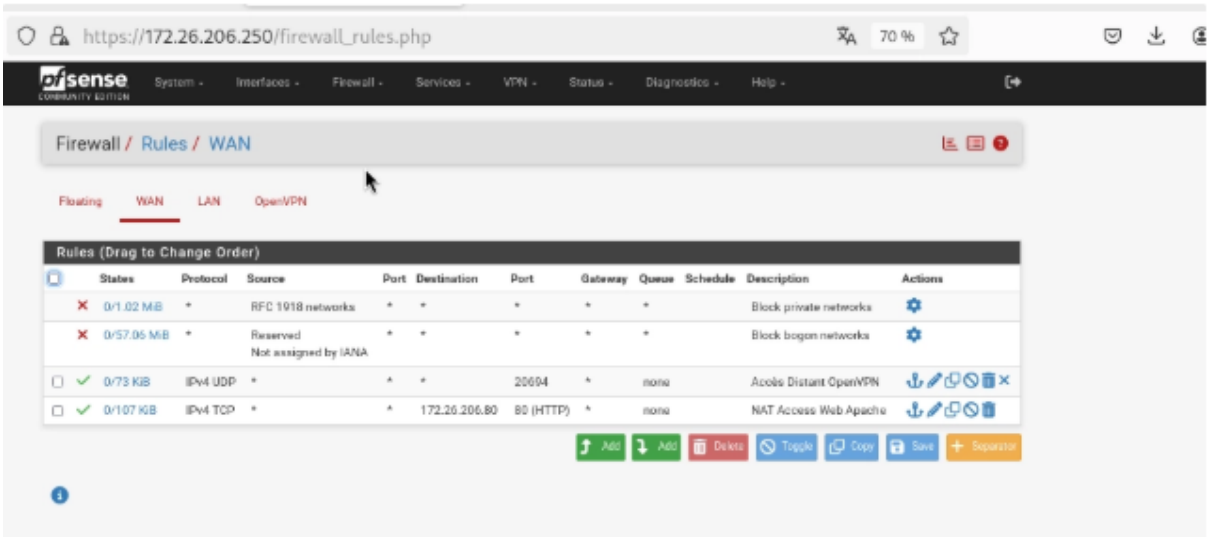
J'ai sélectionné l'interface WAN pour intercepter le trafic entrant.

J'ai choisi le protocole TCP, utilisé par le service web.

Le flux arrivant sur l'adresse IP du WAN est redirigé.

J'ai redirigé ce trafic vers l'adresse interne de mon serveur, 172.26.206.80, sur le port 80 (HTTP).

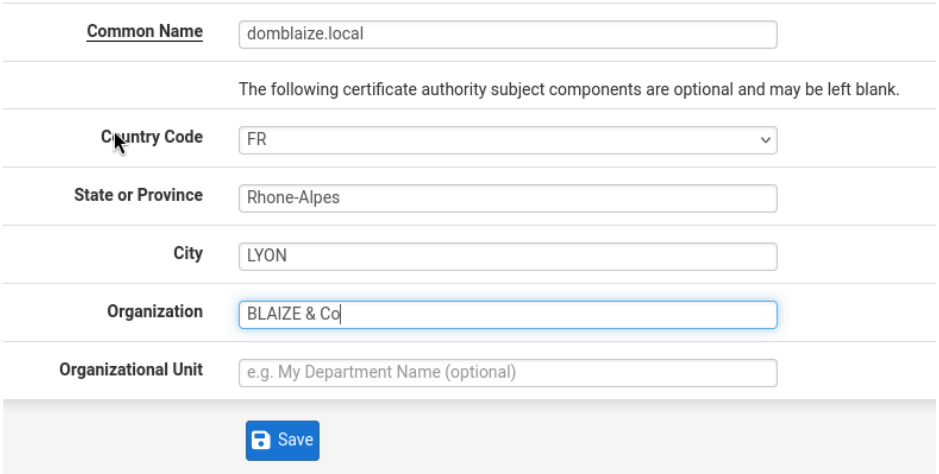
Cette étape me permet de valider que mon serveur Apache est fonctionnel et joignable via l'adresse publique de ma box avant même d'utiliser le tunnel VPN. Elle complète la redirection de port 20694 nécessaire au fonctionnement du serveur OpenVPN lui-même



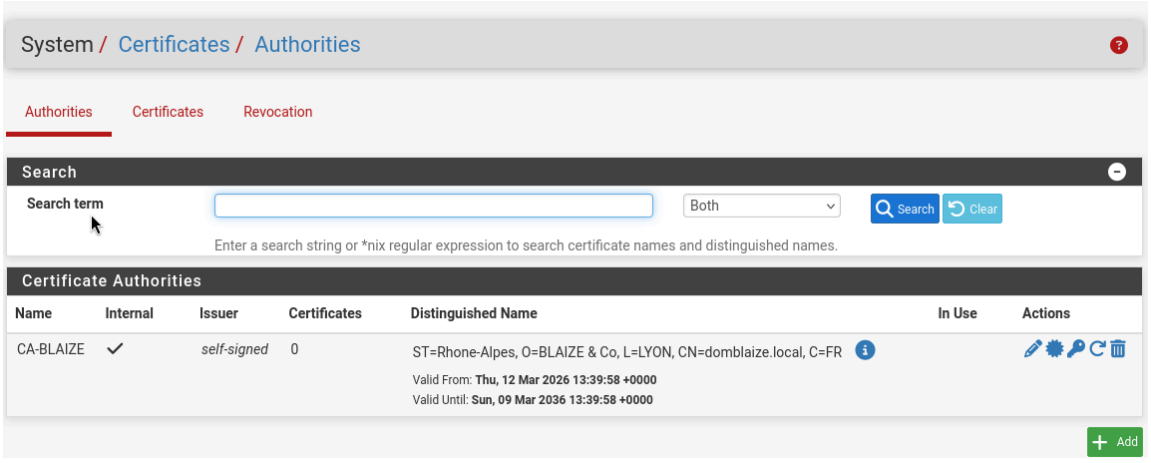
J'ai ouvert le port UDP 20694 pour permettre aux clients VPN externes d'établir la connexion avec le serveur.
J'ai autorisé le trafic TCP sur le port 80 vers l'adresse interne 172.26.206.80, ce qui correspond à la règle de redirection de port (NAT) configurée précédemment.

DEUXIÈME PARTIE

1 - Créer l'autorité de certification

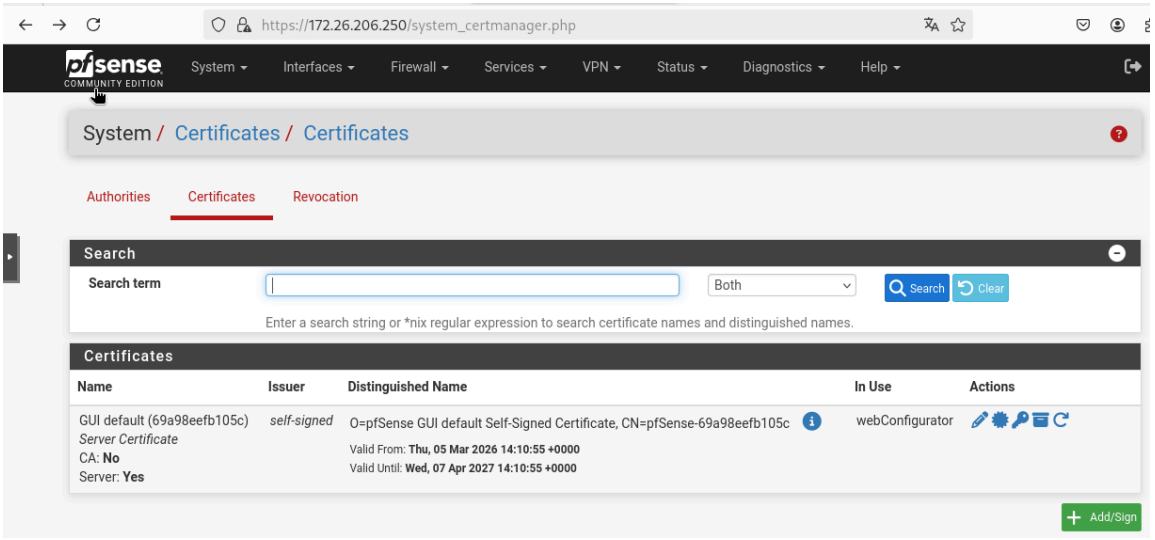


J'ai rempli les champs nécessaires pour définir l'identité de ma propre autorité de certification (CA). J'ai choisi domblaize.local comme Common Name et j'ai renseigné les informations de localisation : France (FR), Rhone-Alpes et LYON. J'ai également nommé mon organisation BLAIZE & Co. J'ai ensuite cliqué sur le bouton Save pour valider la création de cette racine de confiance.



Je regarde maintenant le résultat de l'étape précédente dans l'onglet Authorities (Autorités) du menu System / Certificates.
Je confirme que l'autorité de certification nommée « CA-BLAIZE » a bien été générée. Je peux voir dans les détails techniques qu'elle est marquée comme « Internal » (car je l'ai créée localement) et « self-signed » (auto-signée).
Concernant sa validité, je note qu'elle est active depuis aujourd'hui, le 12 mars 2026. Le système affiche son « Distinguished Name » complet, ce qui me permet de vérifier qu'il reprend exactement les informations que j'ai saisies juste avant.

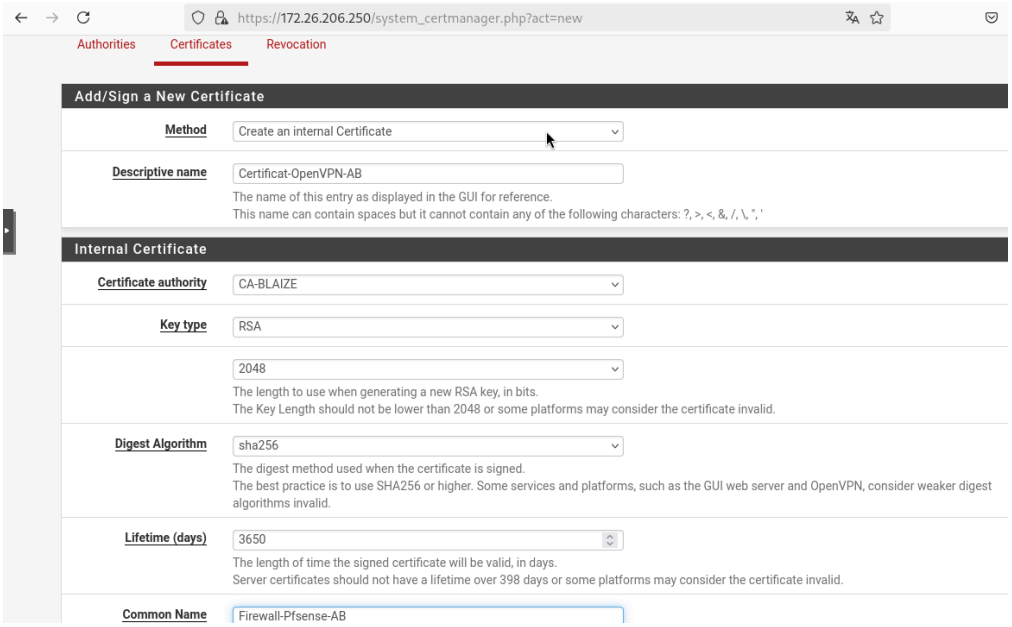
2 - Créer un certificat pour le serveur



Je me suis rendu dans l'onglet **Certificates** (Certificats) pour voir ce qui est déjà installé.

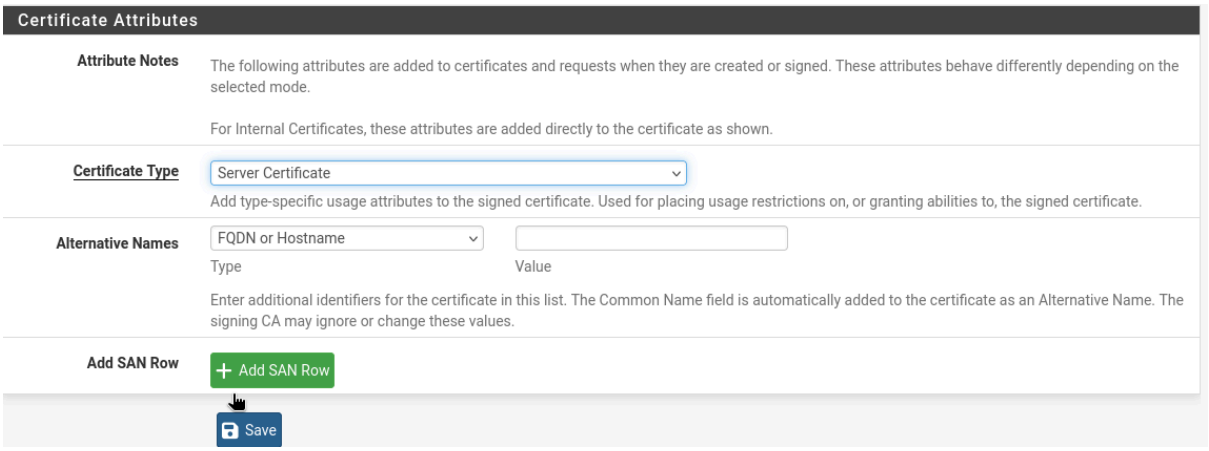
Il n'existe pour l'instant qu'un seul certificat, le **« GUI default »**, qui est auto-signé par le système lui-même pour l'accès à l'interface web.

J'ai cliqué sur le bouton vert **+ Add/Sign** en bas à droite pour lancer la création d'un nouveau certificat qui, cette fois, sera officiellement signé par mon autorité de certification « CA-BLAIZE ».



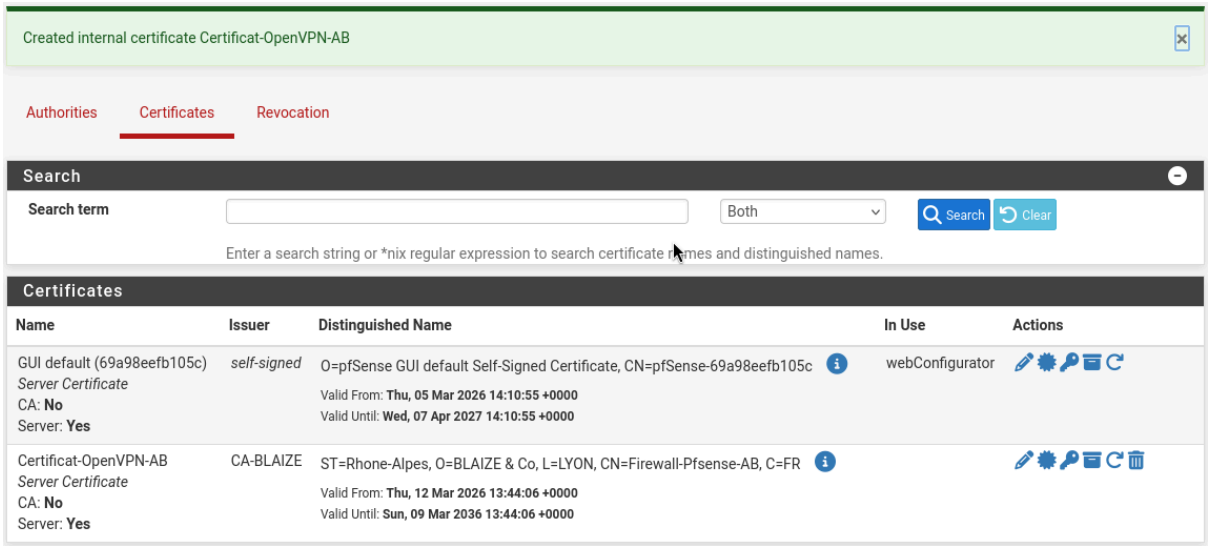
J'ai entamé la configuration du formulaire pour générer ce certificat, qui servira à sécuriser mes accès distants. Pour la méthode, j'ai sélectionné « Create an internal Certificate » afin que la gestion soit centralisée sur pfSense.

J'ai défini l'identité du certificat en le nommant « Certificat-OpenVPN-AB » dans la description pour le repérer facilement, tout en précisant « Firewall-Pfsense-AB » dans le champ Common Name. Pour établir la chaîne de confiance, j'ai bien rattaché ce certificat à mon autorité « CA-BLAIZE », ce qui lui donne sa légitimité au sein de mon réseau. Concernant le chiffrement, je suis parti sur du RSA 2048 bits avec l'algorithme SHA256, et j'ai réglé la durée de validité sur 3650 jours, soit une dizaine d'années, pour assurer la continuité du service sans avoir à le renouveler rapidement.



J'ai défini le **Certificate Type** sur « Server Certificate », ce qui est indispensable pour que le service (comme OpenVPN ou l'interface web du pare-feu) puisse l'utiliser pour prouver son identité aux clients.. Enfin, je m'apprête à cliquer sur **Save** pour finaliser la génération du certificat et l'enregistrer dans ma base locale.

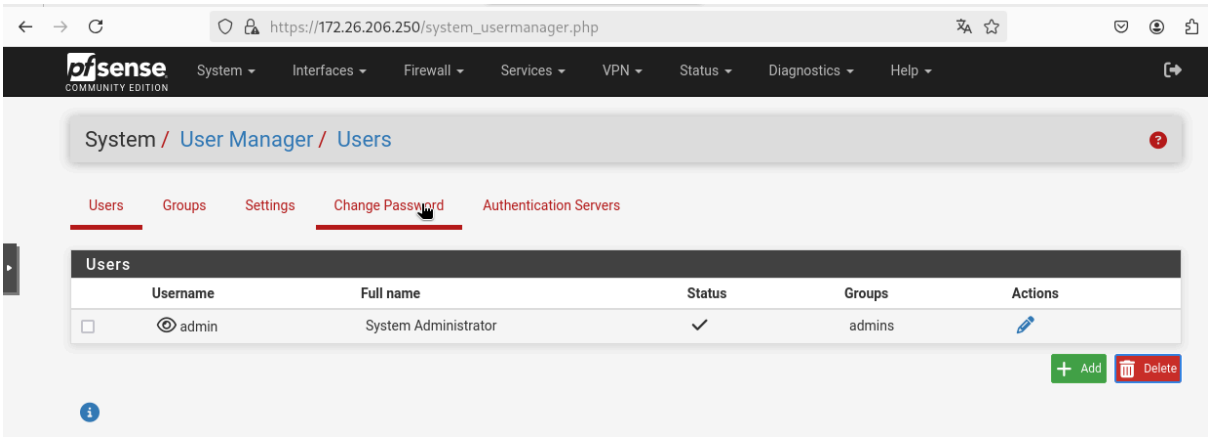
En validant cette étape, je dispose maintenant d'un certificat serveur complet, signé par mon autorité de certification.



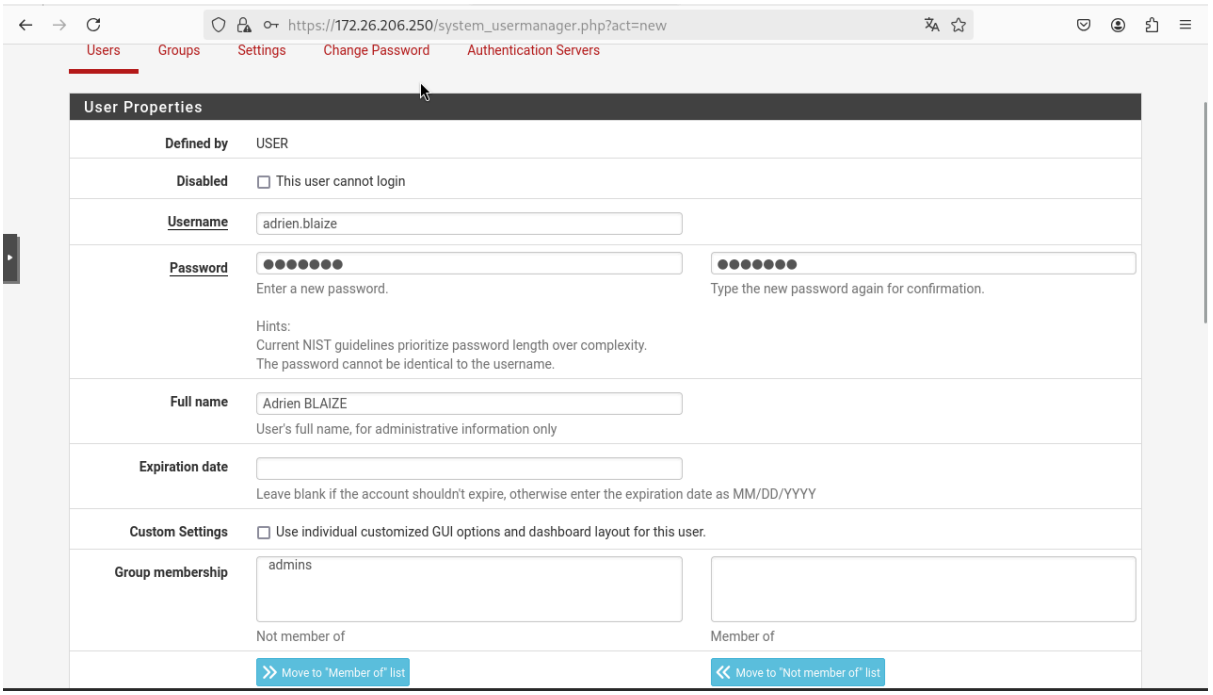
J'ai maintenant la confirmation que la création de mon certificat interne est terminée grâce au bandeau de succès vert en haut de l'écran.

Dans la liste des certificats, je constate que mon nouveau certificat, **« Certificat-OpenVPN-AB »**, est bien présent. Je vérifie que l'émetteur (Issuer) est bien **« CA-BLAIZE »**, ce qui prouve qu'il est correctement rattaché à mon autorité de certification. Les détails confirment qu'il s'agit d'un certificat serveur, qu'il est valide jusqu'au 9 mars 2036, et qu'il reprend l'intégralité des informations d'identité que j'ai saisies précédemment.

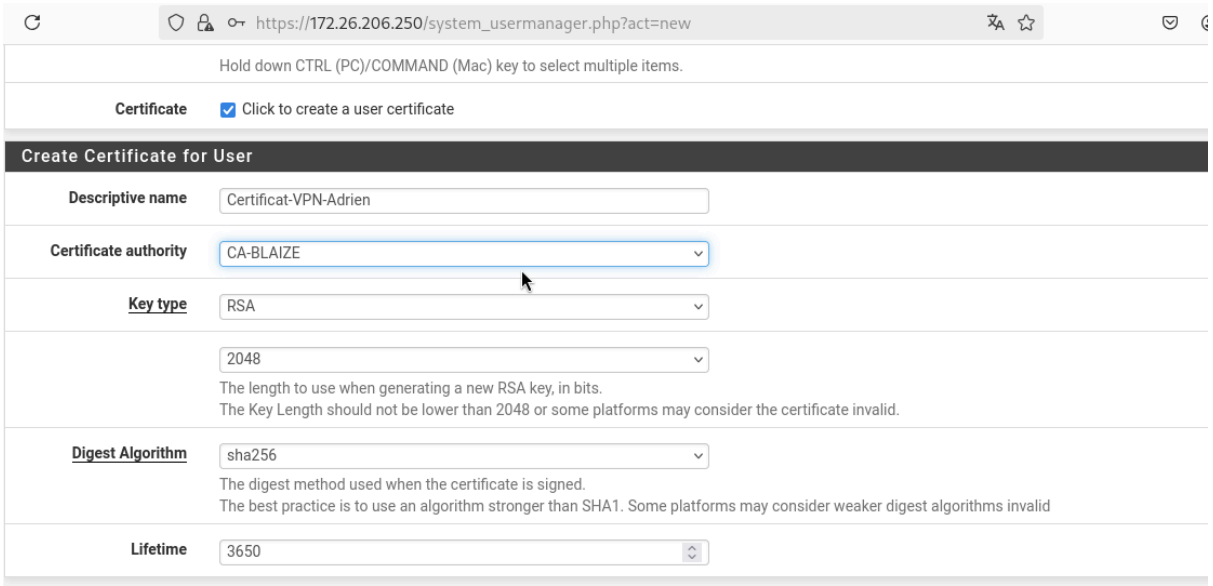
3 - Créer un utilisateur (et son certificat)



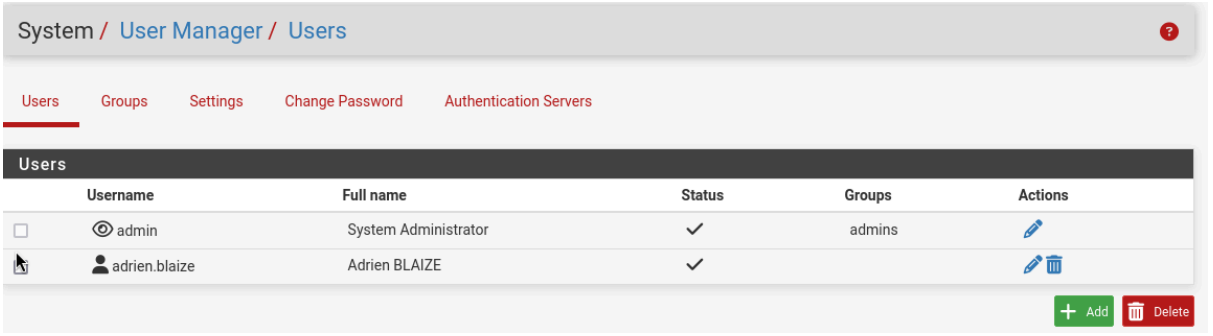
Je me suis rendu dans le menu System / User Manager pour ajouter un nouvel utilisateur (Adrien BLAIZE).



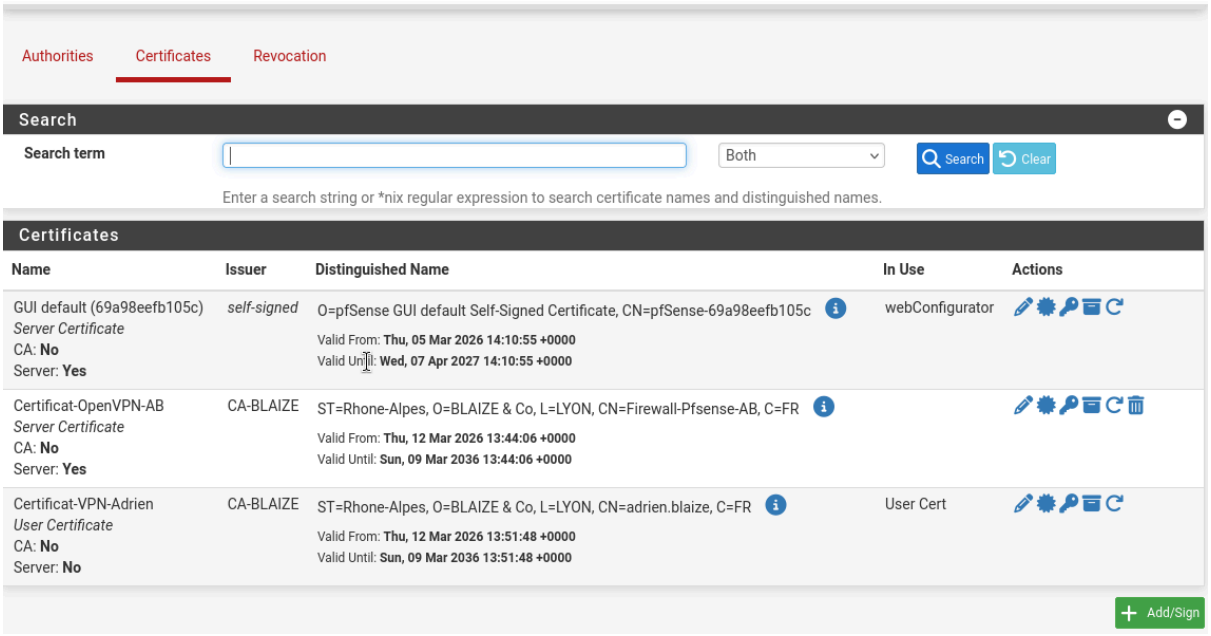
Je suis en train de renseigner les propriétés de Adrien BLAIZE.
J'ai également défini un mot de passe sécurisé pour ce compte. Je prépare ainsi une identité propre servant à la connection au VPN à la fin de la doc.



J'ai coché l'option **« Click to create a user certificate »** pour lier directement un moyen d'authentification à son profil. J'ai nommé ce certificat **« Certificat-VPN-Adrien »** et, en toute logique, je l'ai rattaché à mon autorité de certification **« CA-BLAIZE** J'ai conservé un chiffrement **RSA 2048 bits** avec l'algorithme **SHA256** et une durée de validité de **3650 jours**.



On peut voir que le compte a été créé avec succès.



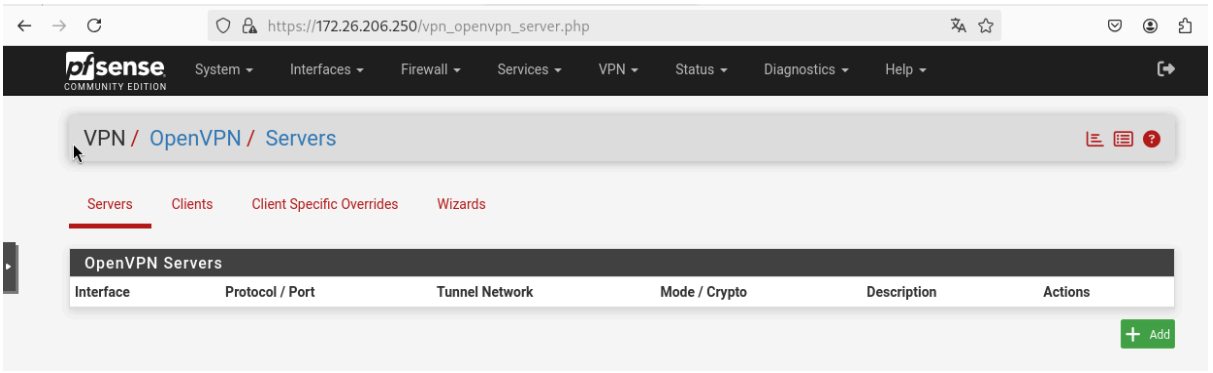
Je suis revenu dans l'onglet **Certificates** pour faire un point sur l'état actuel de ma base de données.

Je constate que la liste est maintenant complète avec trois entrées distinctes :

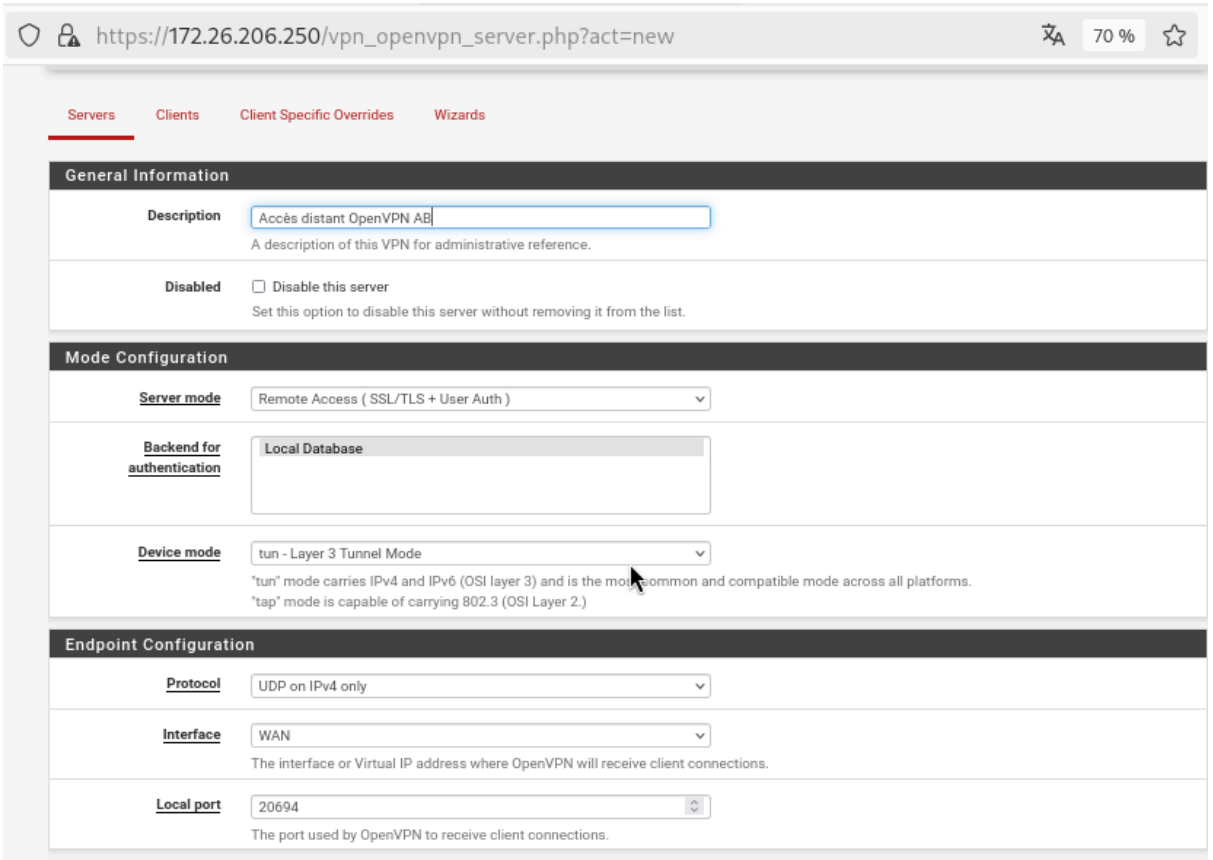
- Le certificat par défaut (**GUI default**).
- Le certificat serveur (**Certificat-OpenVPN-AB**) que j'ai généré plus tôt.
- Le certificat utilisateur (**Certificat-VPN-Adrien**) que je viens de créer pour mon compte d'Adrien.

Je remarque que ce nouveau certificat est bien rattaché à mon autorité **CA-BLAIZE** et qu'il est déjà identifié comme **« User Cert »** dans la colonne « In Use ». Cela confirme que le lien entre l'utilisateur et son certificat est opérationnel : une autorité de confiance, un certificat pour le serveur et un certificat pour l'utilisateur.

4 - Créer la configuration OpenVPN



Je me suis rendu dans le menu `VPN / OpenVPN / Servers`. Comme la liste des serveurs était vide, j'ai cliqué sur le bouton + Add pour lancer mettre en place mon tunnel d'accès distant.



Je suis en train de définir les réglages du serveur pour la sécurité et l'accessibilité :

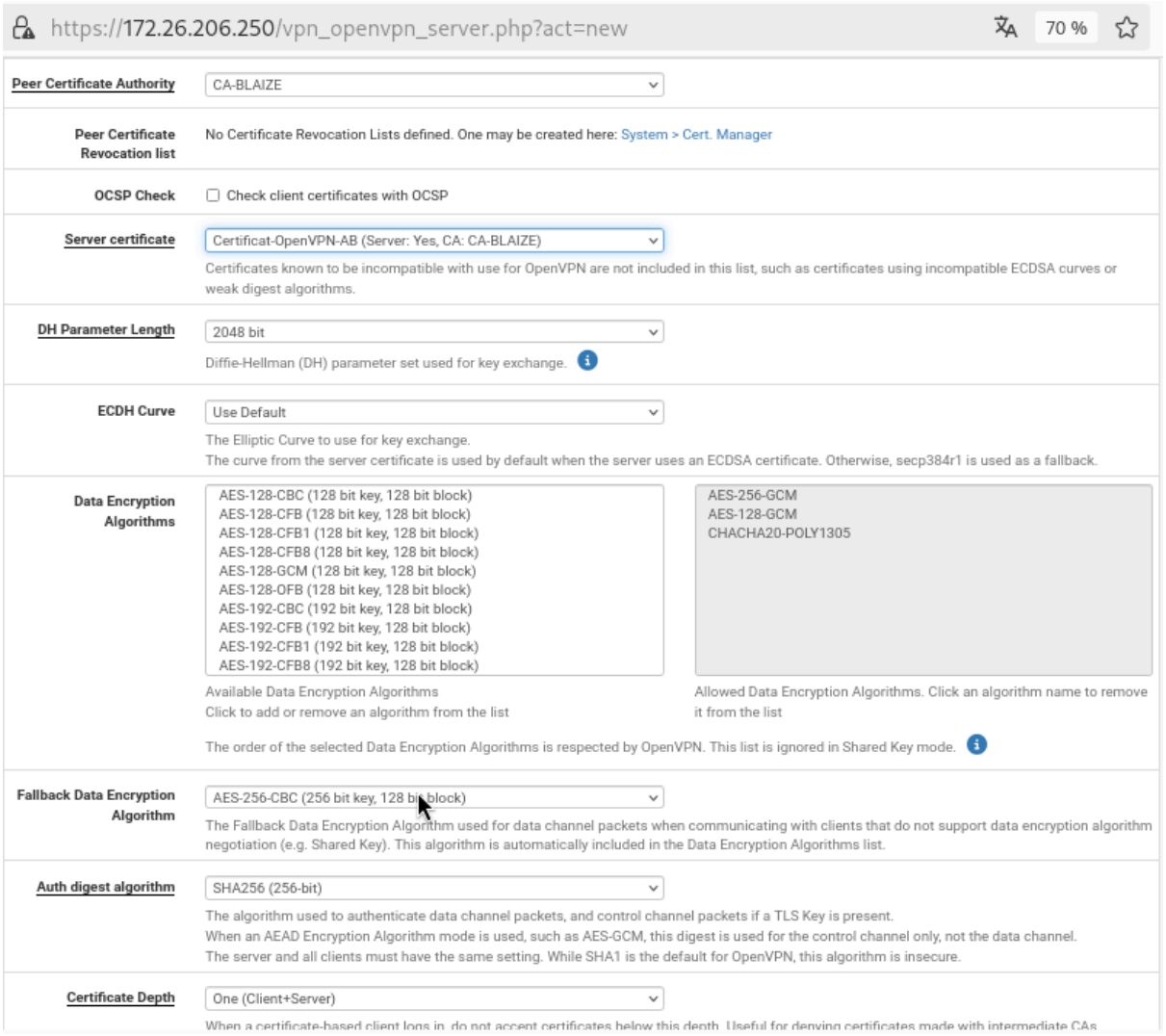
J'ai donné une description, « Accès distant OpenVPN AB », pour identifier ce service dans l'interface.

J'ai opté pour le mode « SSL/TLS + User Auth ». C'est un choix important car cela m'oblige à utiliser à la fois un certificat et un identifiant/mot de passe (celui d'Adrien que j'ai créé juste avant), ce qui renforce nettement la sécurité.

J'ai laissé le mode « tun » (Layer 3) pour le routage IP standard.

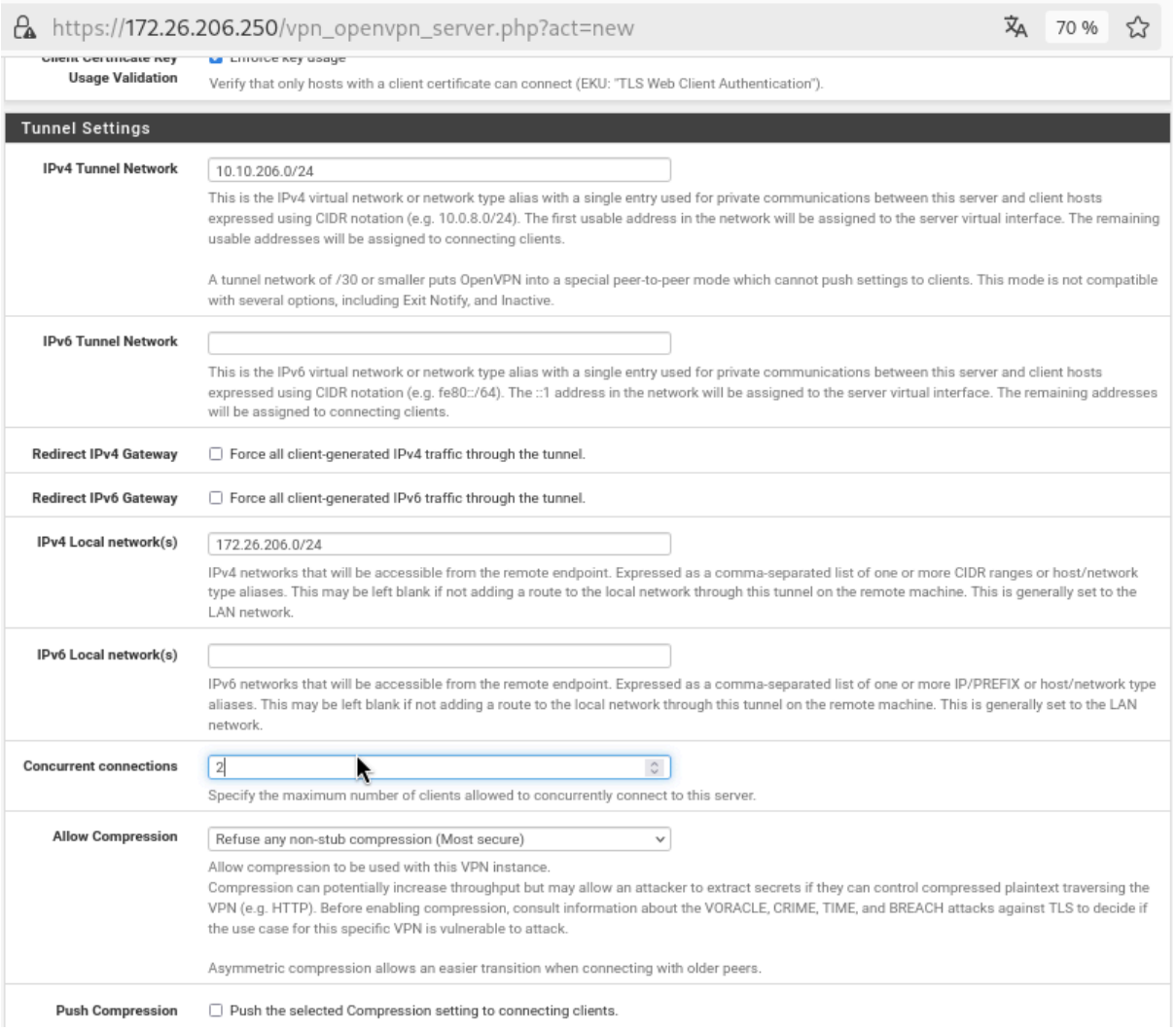
J'ai sélectionné le protocole UDP sur l'interface WAN.

Enfin, j'ai personnalisé le port d'écoute en saisissant 20694, ce qui permet de ne pas utiliser le port par défaut et d'apporter une couche de sécurité

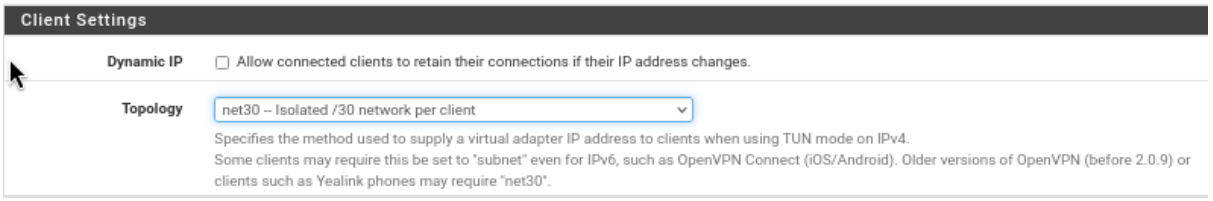


J'ai sélectionné `CA-BLAIZE` comme Peer Certificate Authority. Le serveur sait quelle autorité doit avoir signé les certificats des clients qui tentent de se connecter.
Pour le champ Server certificate, j'ai choisi le certificat `Certificat-OpenVPN-AB` que j'ai généré plus tôt. Le serveur l'utilisera pour prouver son identité aux utilisateurs.
J'ai sélectionné des algorithmes dans la liste des autorisés, notamment le AES-256-GCM et le CHACHA20-POLY1305.

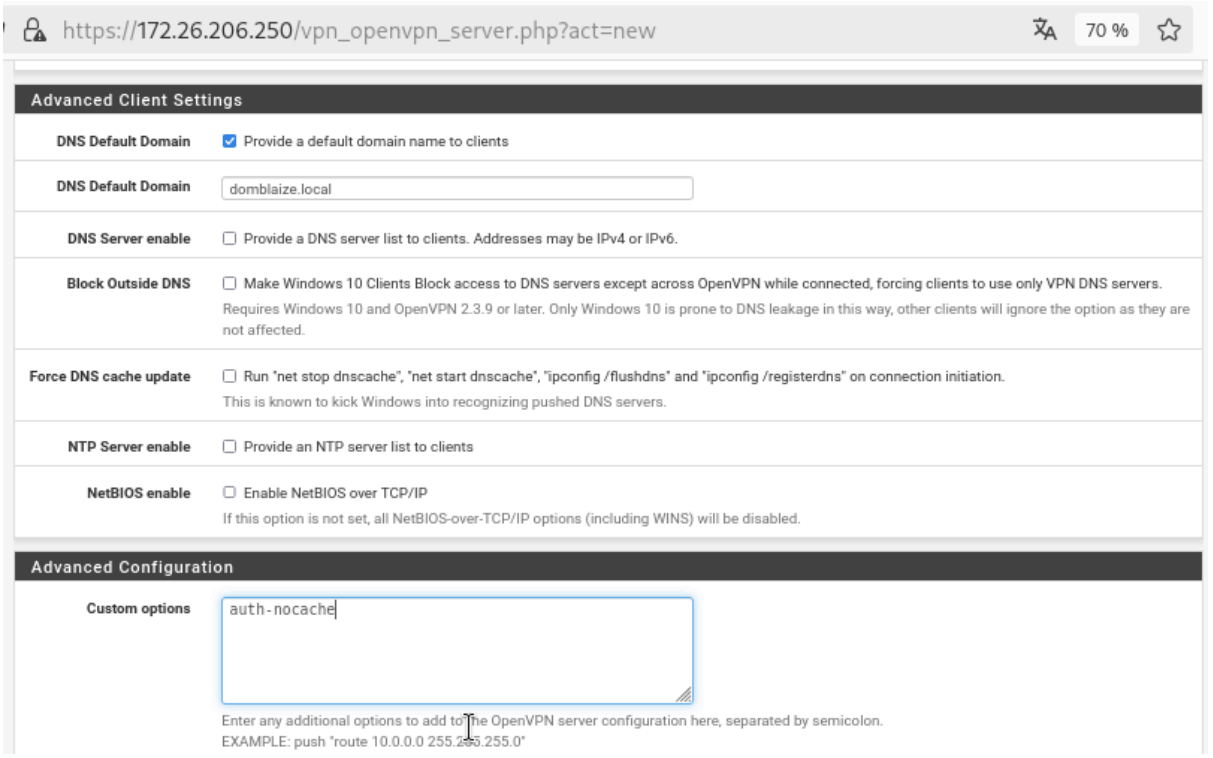
J'ai réglé la longueur des paramètres Diffie-Hellman (DH) sur 2048 bits.
J'ai choisi le SHA256 comme algorithme de hachage pour l'authentification des paquets.
Pour la Certificate Depth, j'ai laissé sur « One (Client+Server) », ce qui suffit largement puisque j'utilise une CA simple sans certificats intermédiaires.



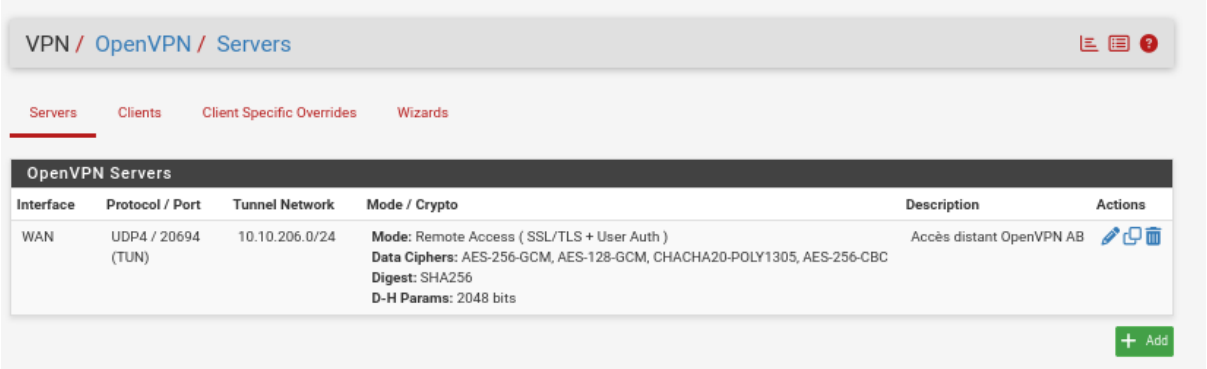
J'ai défini le IPv4 Tunnel Network sur `10.10.206.0/24`. C'est la plage d'adresses qui sera distribuée dynamiquement à mes clients VPN lors de leur connexion.
J'ai renseigné le champ IPv4 Local network(s) avec `172.26.206.0/24`.
C'est une étape importante pour que les utilisateurs distants puissent réellement "voir" et atteindre les machines de mon réseau interne.
J'ai bridé le nombre de Concurrent connections à 2, ce qui suffit largement pour mes tests actuels.
Pour éviter les vulnérabilités comme les attaques VORACL, j'ai laissé le choix de refuser la compression dans le champ Allow Compression.



J'ai gardé la topologie net30, qui est un standard permettant d'isoler chaque client dans son propre petit sous-réseau virtuel.



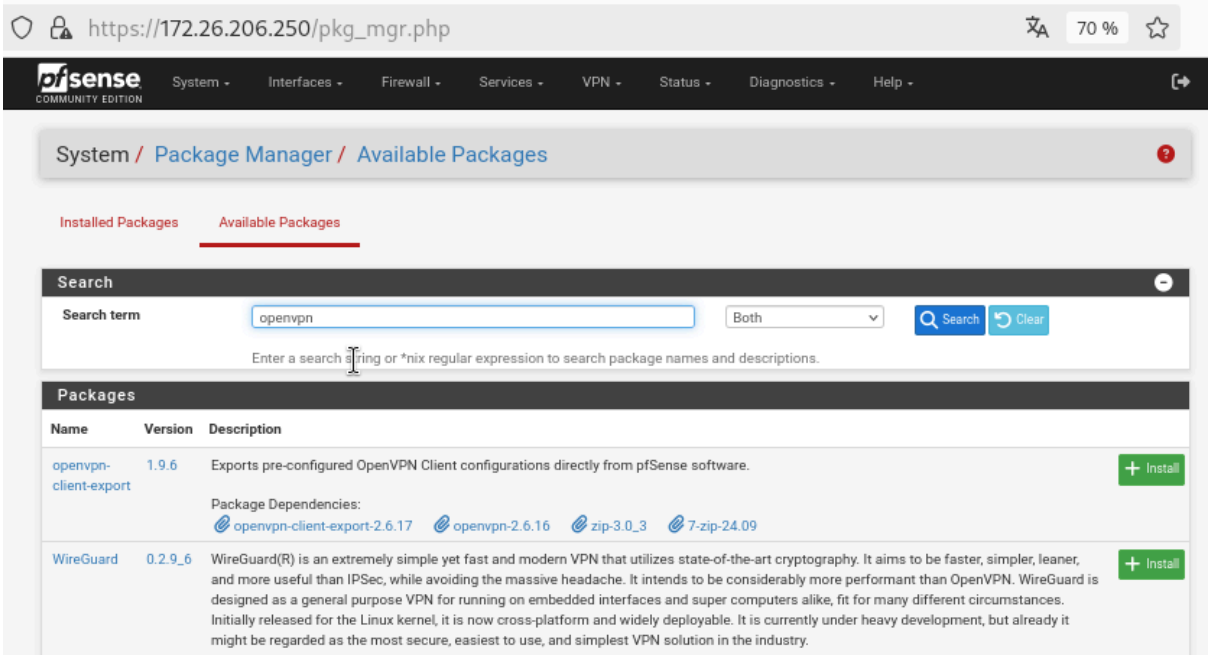
J'ai activé l'option DNS Default Domain et j'ai renseigné `domblaize.local`. Ça permettra aux clients connectés de résoudre les noms d'hôtes de mon réseau local sans avoir à saisir le suffixe complet à chaque fois.
Dans la section Custom options, j'ai ajouté la directive `auth-nocache`. C'est une mesure de sécurité importante pour éviter que les identifiants ne restent stockés en clair dans la mémoire vive (RAM) du client après la connexion.



Une fois que j'ai cliqué sur Save, je suis revenu sur la page principale de l'onglet Servers.

Je vois maintenant mon serveur « Accès distant OpenVPN AB » dans la liste.
Le tableau récapitule bien ma config : il écoute sur l'interface WAN, en UDP sur le port 20694. On voit aussi que le tunnel network en `10.10.206.0/24` est opérationnel et que le chiffrement fort (AES-256-GCM, SHA256) est bien appliqué.

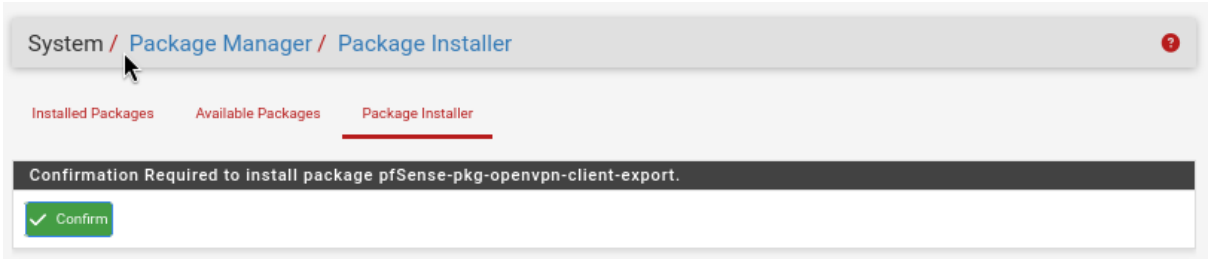
5 - Exporter la configuration OpenVPN



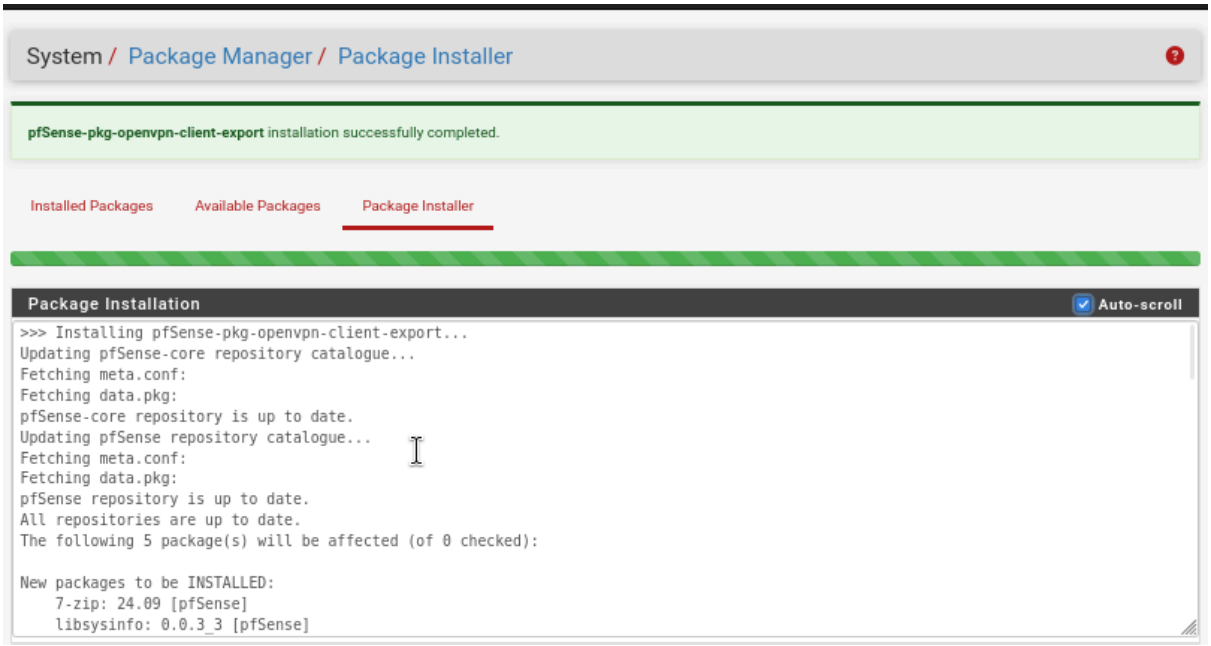
Dans l'onglet Available Packages, j'ai lancé une recherche avec le mot-clé « openvpn » pour trouver les outils associés.

J'ai identifié le paquet `openvpn-client-export`. C'est un outil indispensable car il permet de générer automatiquement les fichiers de configuration (.ovpn) et les installateurs Windows/macOS pour les utilisateurs. Sans lui, je devrais tout configurer à la main sur chaque poste client.

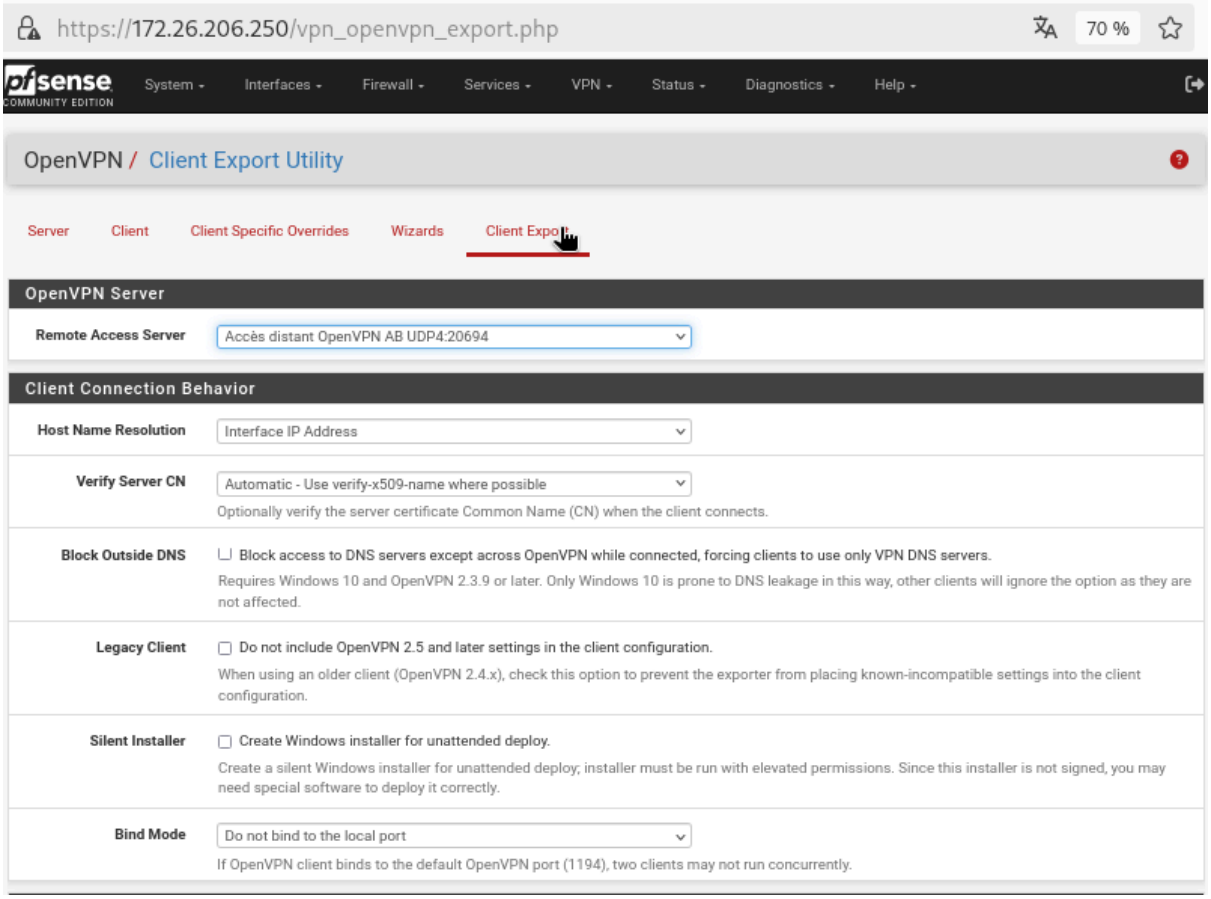
Je m'apprête à cliquer sur le bouton + Install pour l'ajouter à mon pfSense.



Je clique sur confirmer.

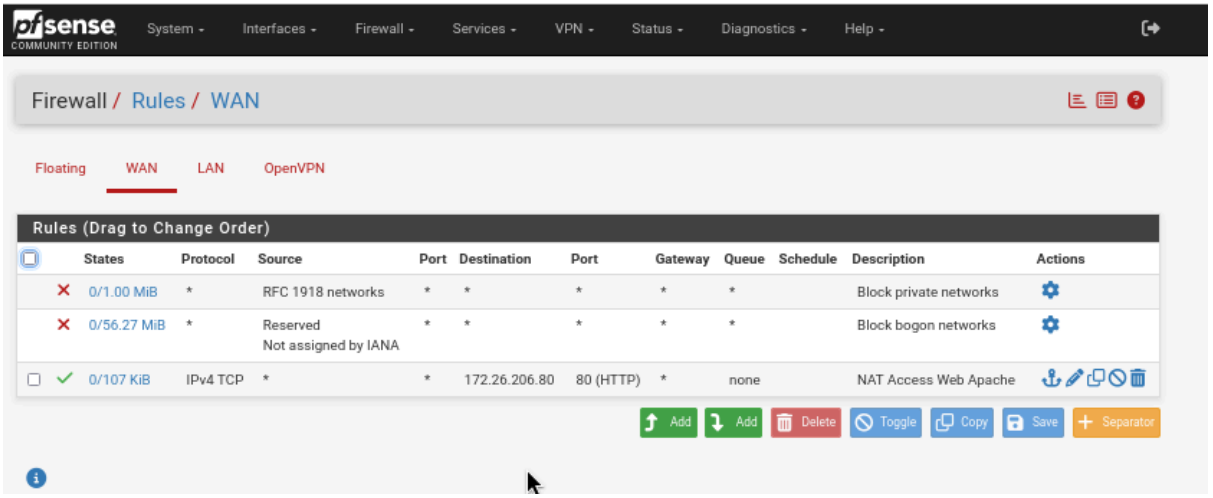


J'ai terminé l'installation du paquet `openvpn-client-export`. Le bandeau vert en haut de la page confirme que l'opération a réussi. Grâce à cette étape, j'ai maintenant accès aux outils qui me permettent de créer les installateurs et les fichiers `.ovpn` de manière automatisée, ce qui m'évite de configurer chaque client manuellement.



J'ai bien ciblé mon serveur « Accès distant OpenVPN AB » qui utilise le port 20694.

Pour le champ Host Name Resolution, j'ai choisi « Interface IP Address ». C'est pour que le fichier de configuration indique au client de se connecter directement à l'adresse IP de mon interface WAN.
J'ai laissé la vérification du CN (Common Name) du serveur sur « Automatic » pour m'assurer que la validation du certificat se fasse correctement lors de l'établissement du tunnel.



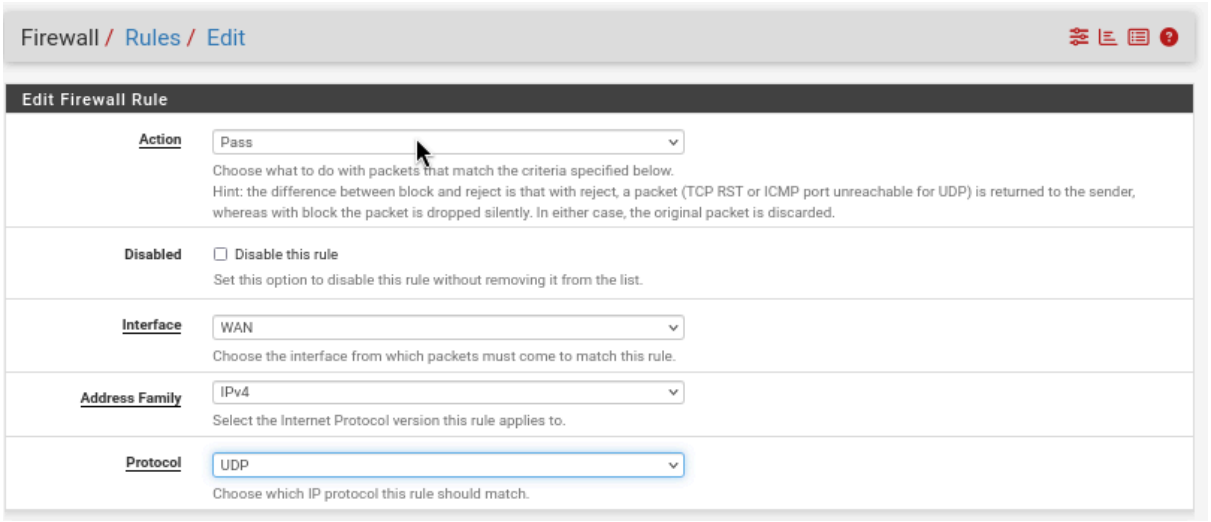
J'ai analysé les règles déjà actives sur mon interface :

Je retrouve les blocages de sécurité standards (réseaux privés et réseaux "bogons").

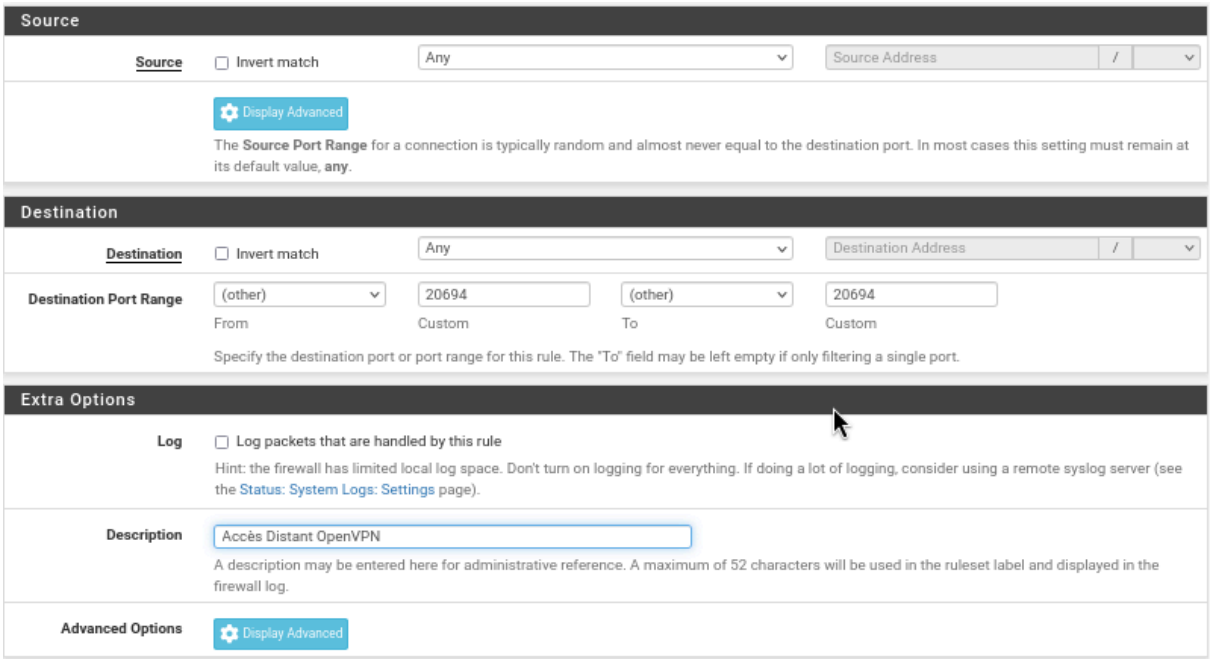
Il existe déjà une règle de NAT pour laisser passer le trafic vers un serveur Web Apache sur l'IP `172.26.206.80`.

Cependant, je constate que mon serveur OpenVPN n'est pas encore autorisé.
Je m'apprête donc à cliquer sur le bouton Add pour créer la règle d'exception qui permettra au flux d'arriver sur le port 20694 en UDP.
Sans elle, le pare-feu bloquera systématiquement les tentatives de connexion, même si les certificats de mon compte sont valables.

6 - Créer les règles de Pare-feu

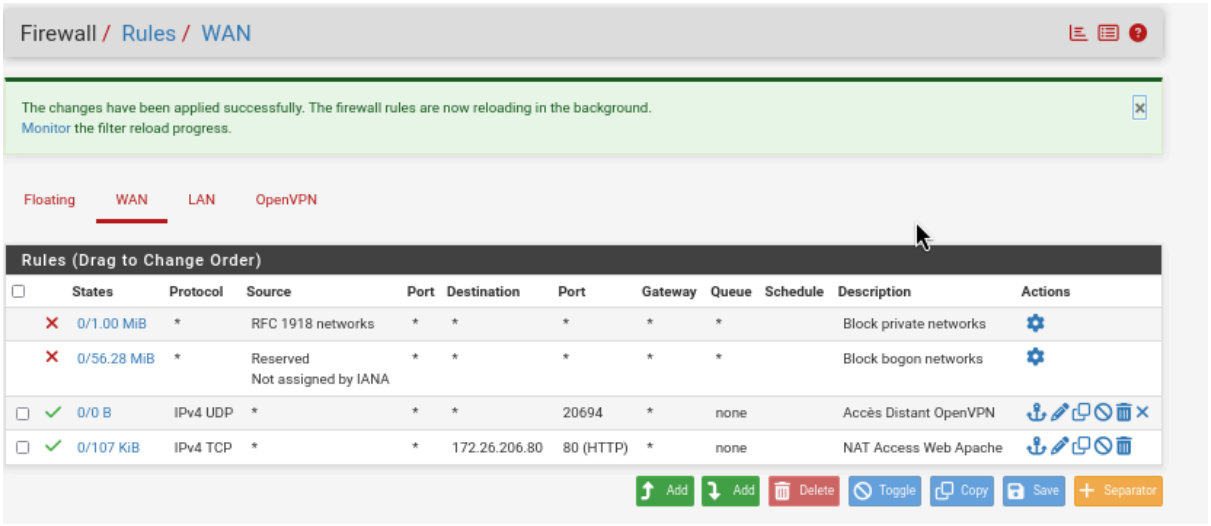


Je suis entré dans l'édition d'une nouvelle règle. J'ai défini l'action sur « Pass » pour autoriser le flux. Comme mon serveur OpenVPN utilise le protocole UDP sur IPv4, j'ai sélectionné ces paramètres pour ne pas ouvrir plus que nécessaire.

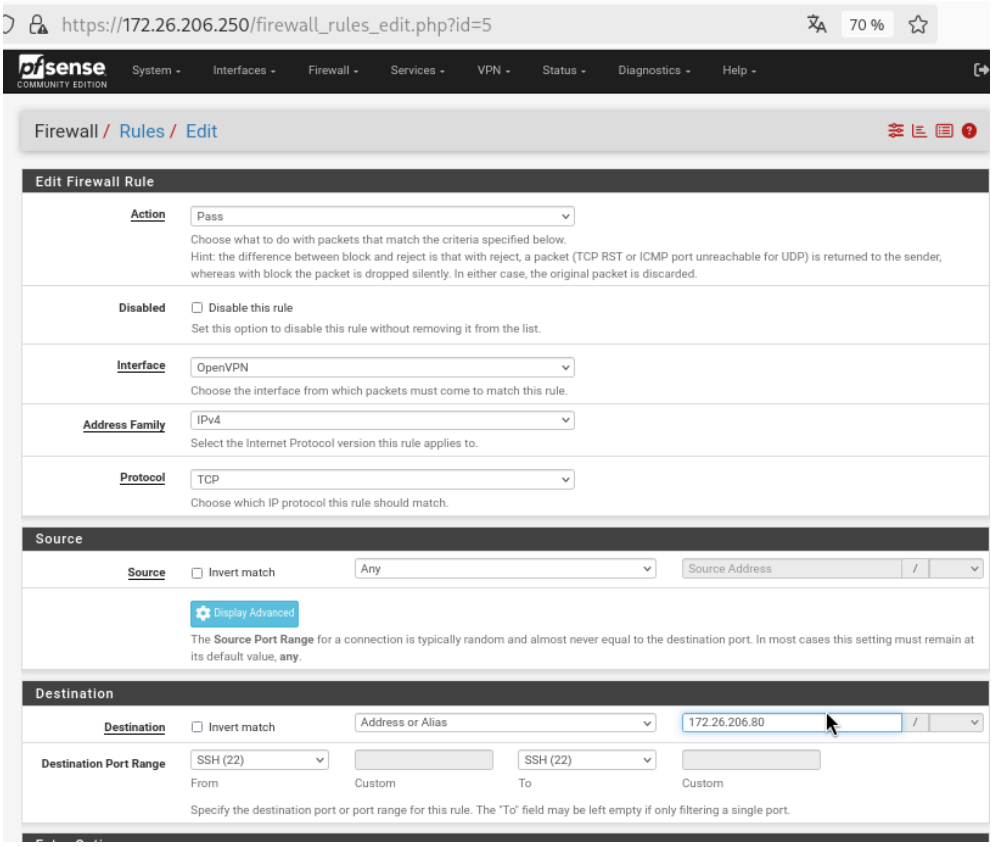


J'ai configuré les points d'entrée et de sortie du flux :

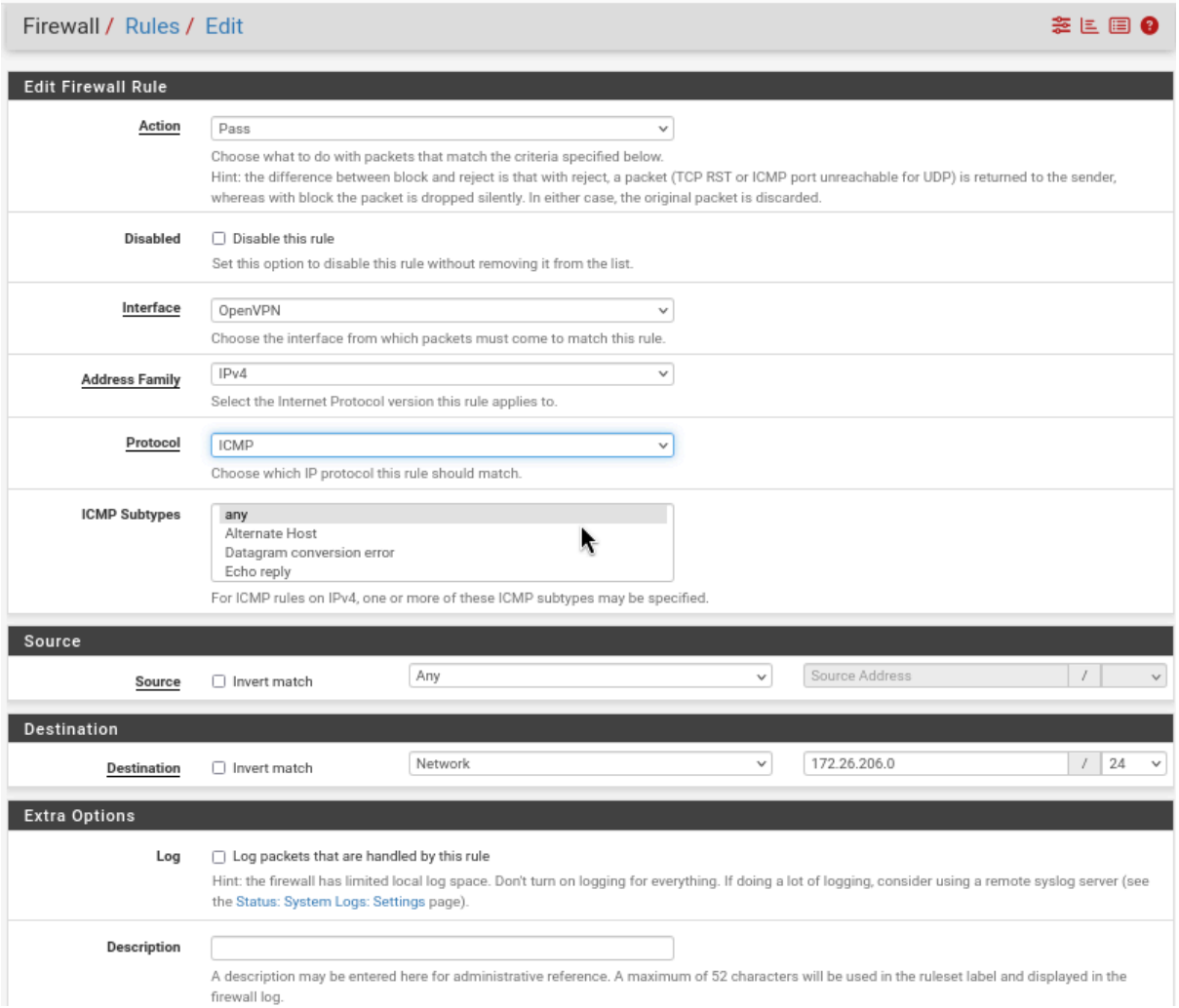
J'ai laissé sur « Any » puisque, par définition,on souhaite se connecter de partout.
J'ai spécifié le port personnalisé 20694 dans les deux champs (From/To) du Destination Port Range.
J'ai rempli le champ Description avec « Accès Distant OpenVPN ».



Après avoir sauvegardé, j'ai cliqué sur le bouton pour appliquer les changements. Le bandeau vert me confirme que les règles sont bien rechargées. Dans mon tableau récapitulatif du WAN, je vois maintenant ma nouvelle règle active (avec la coche verte). Elle est bien positionnée pour laisser passer le trafic UDP sur le port 20694.



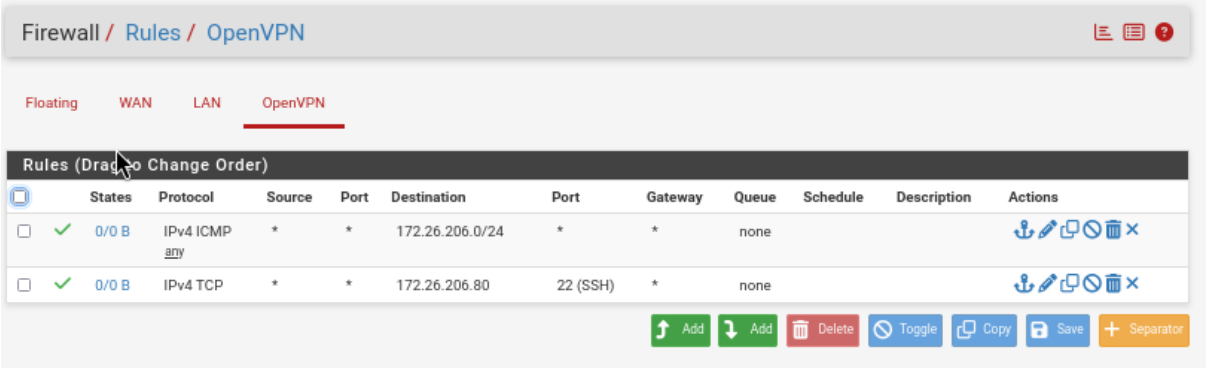
J'ai sélectionné l'interface OpenVPN. C'est ici que je gère ce que les utilisateurs (comme Adrien) ont le droit de faire une fois qu'ils sont connectés au VPN.
J'ai choisi l'action « Pass » avec le protocole TCP.
Plutôt que d'ouvrir l'accès à tout mon réseau, j'ai restreint la règle à une seule machine. J'ai ciblé l'adresse IP 172.26.206.80 (celle où se trouve mon serveur Apache).
J'ai ouvert spécifiquement le port 22 (SSH).



J'ai sélectionné le protocole ICMP (IPv4) avec l'option « any » pour les sous-types. Cela m'autorise notamment à utiliser la commande « ping ».

Contrairement à la règle précédente qui était limitée à un seul serveur, j'ai ici élargi la destination à l'ensemble de mon sous-réseau local : 172.26.206.0/24.

Cette règle est essentielle pour ma phase de test. Elle me permet de vérifier la connectivité de bout en bout en "pingant" n'importe quelle machine du réseau LAN depuis mon accès distant. Si une machine ne répond pas, je saurai que le problème vient potentiellement de son propre pare-feu et non du tunnel VPN lui-même.

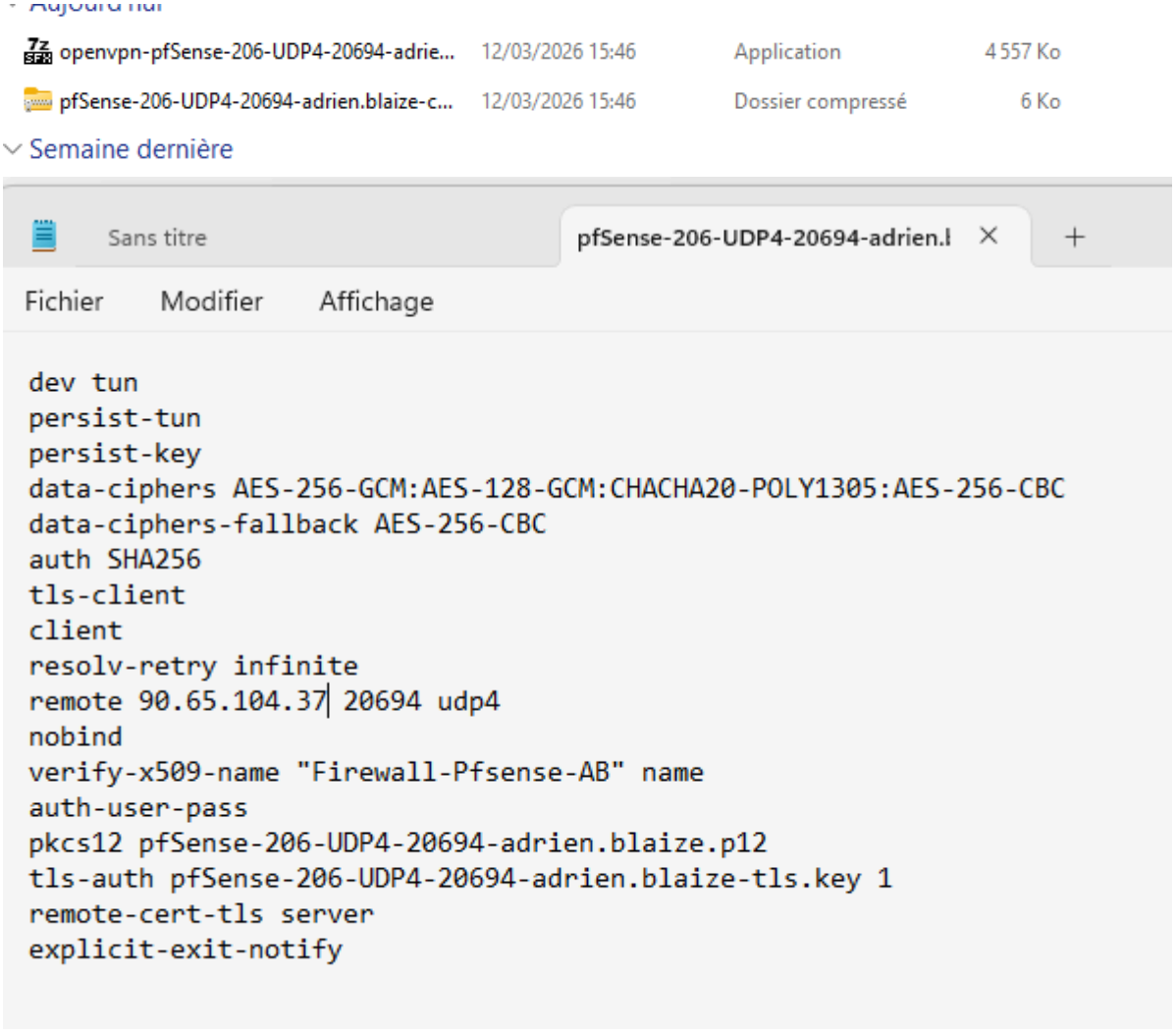


Je peux confirmer que mes deux règles sont bien actives et correctement ordonnées :

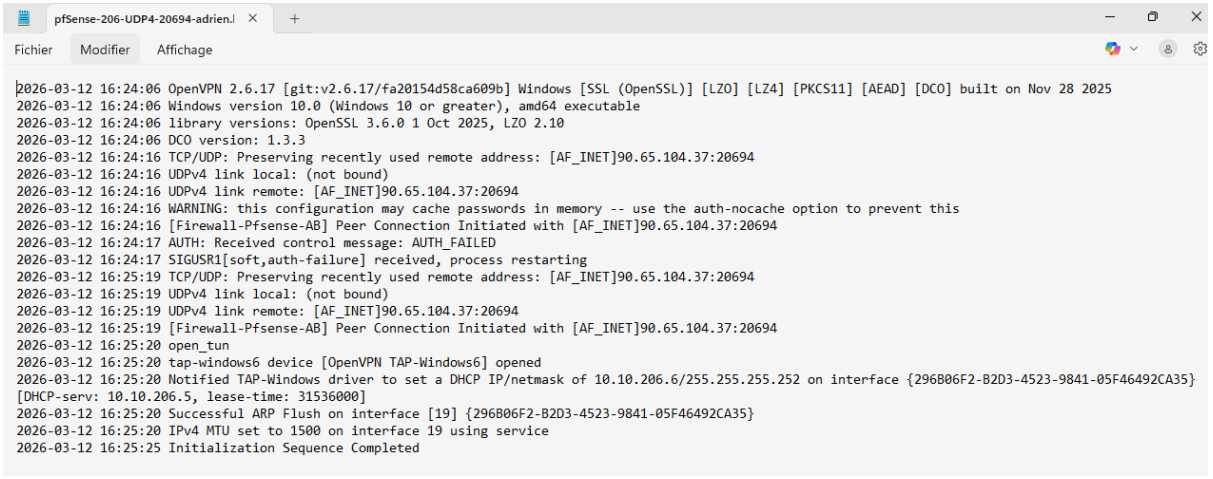
J'ai autorisé le protocole ICMP (le ping) depuis n'importe quelle source du tunnel vers tout le réseau local `172.26.206.0/24`. C'est ma règle de base pour m'assurer que le routage fonctionne bien.

J'ai également activé l'accès TCP sur le port 22 (SSH), mais de manière plus restrictive, uniquement vers la destination `172.26.206.80`

7 - Tester



Sur cette image, j'ai récupéré les fichiers générés par l'utilitaire d'exportation.
Récupération des fichiers : Je vois dans mon dossier de téléchargements l'installeur exécutable et le dossier compressé contenant le profil d'Adrien.
Inspection de la configuration : J'ai ouvert le fichier .ovpn avec le Bloc-notes pour vérifier les directives de connexion. Je confirme que le client pointe bien vers l'adresse IP publique 90.65.104.37 sur le port personnalisé 20694. Je retrouve aussi les paramètres de sécurité que j'avais définis, comme le chiffrement AES-256-GCM et l'authentification par certificat (pkcs12) couplée au login/password (auth-user-pass).

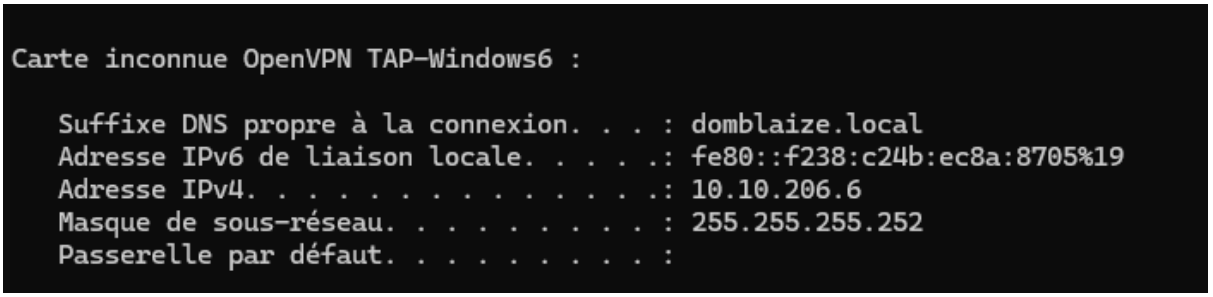


Ici, j'ai lancé la connexion depuis le poste client et j'observe le déroulement de la session dans les logs.

Je remarque qu'une première tentative a échoué avec un message AUTH_FAILED à 16:24:17, dû à une petite erreur de saisie des identifiants.

La seconde tentative est la bonne. Je vois que j’ai bien négocié la couche TLS avec le serveur. Le driver réseau a attribué l'adresse IP virtuelle 10.10.206.6 à ma machine.

La ligne « Initialization Sequence Completed » à 16:25:25 confirme que le tunnel est maintenant monté et opérationnel.



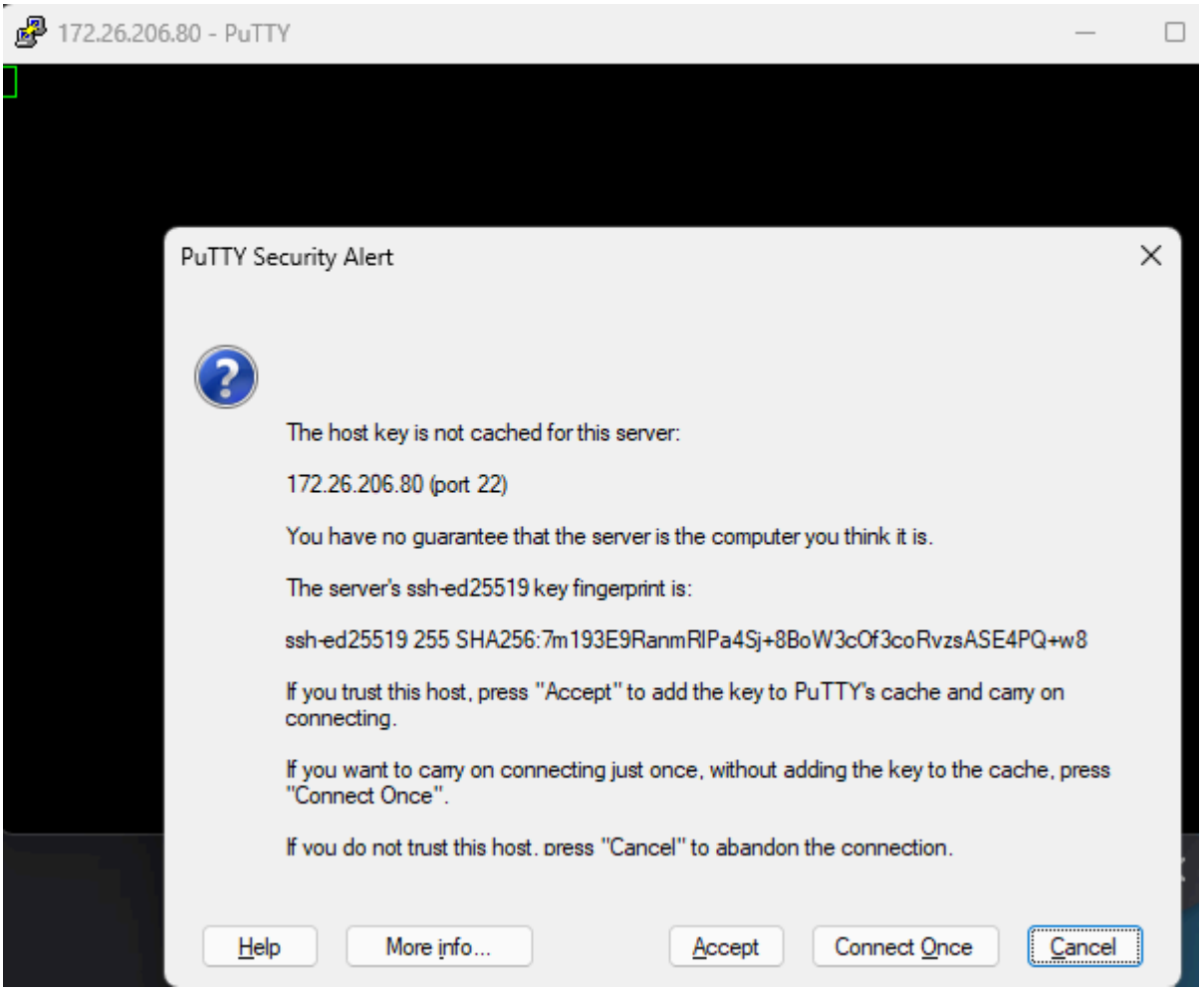
On voit dans le CMD que la carte réseau OpenVPN attribue une adresse ip du réseau interne VPN, prouvant ainsi le bon fonctionnement de celui-ci.

```
IPv4 Table de routage
===========================================================================
Itinéraires actifs :
Destination réseau    Masque réseau  Adr. passerelle   Adr. interface Métrique
          0.0.0.0          0.0.0.0    172.30.11.254    172.30.11.115     25
      10.10.206.1  255.255.255.255      10.10.206.5      10.10.206.6    281
      10.10.206.4  255.255.255.252         On-link       10.10.206.6    281
      10.10.206.6  255.255.255.255         On-link       10.10.206.6    281
      10.10.206.7  255.255.255.255         On-link       10.10.206.6    281
        127.0.0.0        255.0.0.0         On-link         127.0.0.1    331
        127.0.0.1  255.255.255.255         On-link         127.0.0.1    331
  127.255.255.255  255.255.255.255         On-link         127.0.0.1    331
     172.26.206.0    255.255.255.0      10.10.206.5      10.10.206.6    281
     172.30.11.0     255.255.255.0         On-link     172.30.11.115    281
    172.30.11.115  255.255.255.255         On-link     172.30.11.115    281
    172.30.11.255  255.255.255.255         On-link     172.30.11.115    281
     192.168.23.0    255.255.255.0         On-link      192.168.23.1    291
     192.168.23.1  255.255.255.255         On-link      192.168.23.1    291
   192.168.23.255  255.255.255.255         On-link      192.168.23.1    291
    192.168.184.0    255.255.255.0         On-link     192.168.184.1    291
    192.168.184.1  255.255.255.255         On-link     192.168.184.1    291
  192.168.184.255  255.255.255.255         On-link     192.168.184.1    291
        224.0.0.0        240.0.0.0         On-link         127.0.0.1    331
        224.0.0.0        240.0.0.0         On-link     192.168.184.1    291
        224.0.0.0        240.0.0.0         On-link      192.168.23.1    291
        224.0.0.0        240.0.0.0         On-link     172.30.11.115    281
        224.0.0.0        240.0.0.0         On-link       10.10.206.6    281
  255.255.255.255  255.255.255.255         On-link         127.0.0.1    331
  255.255.255.255  255.255.255.255         On-link     192.168.184.1    291
```

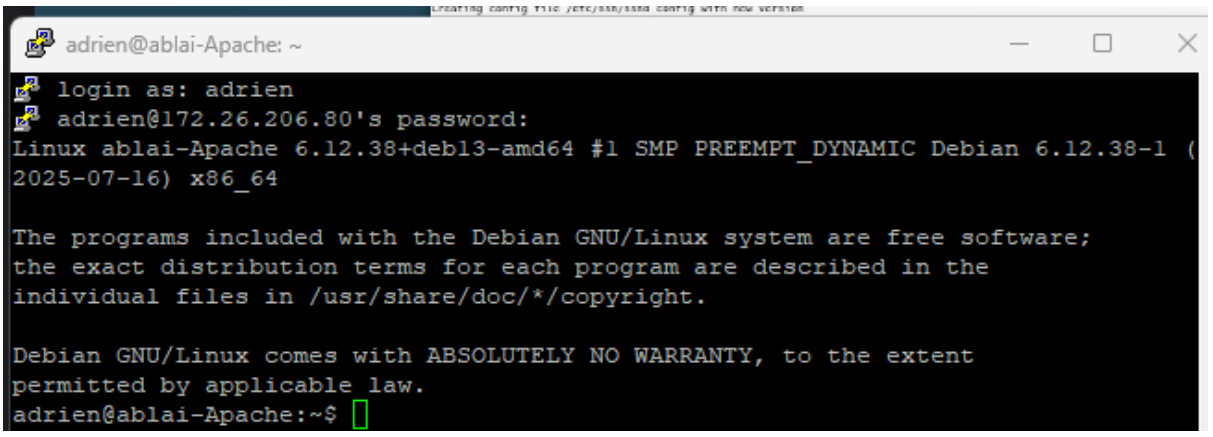
J'ai affiché la table de routage IPv4 sur mon poste Windows pour m'assurer que le tunnel OpenVPN redirige bien le trafic vers mon réseau distant.
Voici ce que j'ai analysé sur cette capture :
Je vois bien mon adresse IP de tunnel, la 10.10.206.6, apparaître dans la colonne « Adr. interface ». C'est elle qui gère le trafic chiffré.
C'est le point le plus important. Je constate que la route vers le réseau 172.26.206.0 (masque 255.255.255.0) est bien présente. Elle pointe vers la passerelle 10.10.206.5 (le serveur pfSense à l'autre bout du tunnel).
La présence de cette ligne prouve que l'option « Local Network » que j'avais configurée sur le serveur a bien été poussée (« pushed ») vers mon client. Sans cette route, je ne pourrais jamais atteindre mon serveur Apache ou faire mes pings, car mon ordinateur ne saurait pas qu'il doit passer par le VPN pour joindre ces adresses.

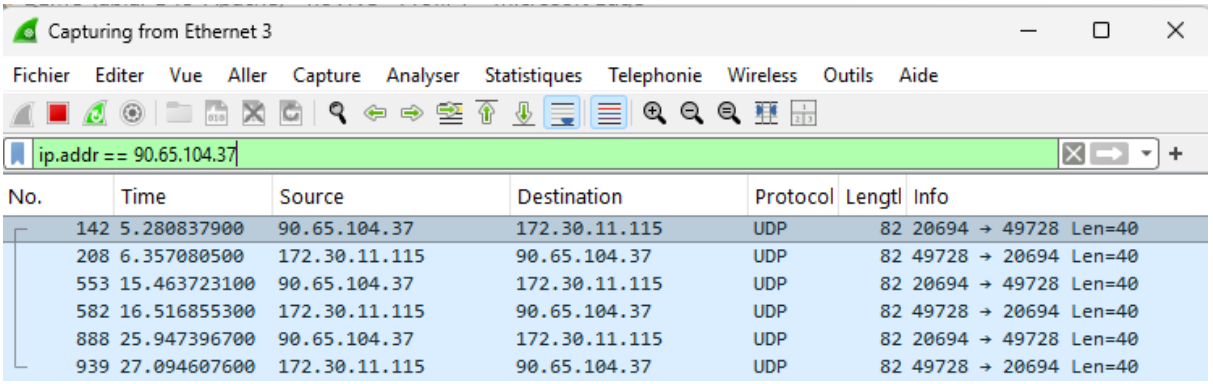
```
PS C:\Users\admin> ping 172.26.206.80

Envoi d'une requête 'Ping'  172.26.206.80 avec 32 octets de données :
Réponse de 172.26.206.80 : octets=32 temps=2 ms TTL=63
Réponse de 172.26.206.80 : octets=32 temps=4 ms TTL=63
Réponse de 172.26.206.80 : octets=32 temps=3 ms TTL=63
Réponse de 172.26.206.80 : octets=32 temps=3 ms TTL=63

Statistiques Ping pour 172.26.206.80:
    Paquets : envoyés = 4, reçus = 4, perdus = 0 (perte 0%),
Durée approximative des boucles en millisecondes :
    Minimum = 2ms, Maximum = 4ms, Moyenne = 3ms
PS C:\Users\admin>
```

Par conséquent, le ping fonctionne logiquement grâce à la règle créée précédemment.



Par conséquent, la connexion par SSH fonctionne logiquement grâce à la règle créée précédemment.

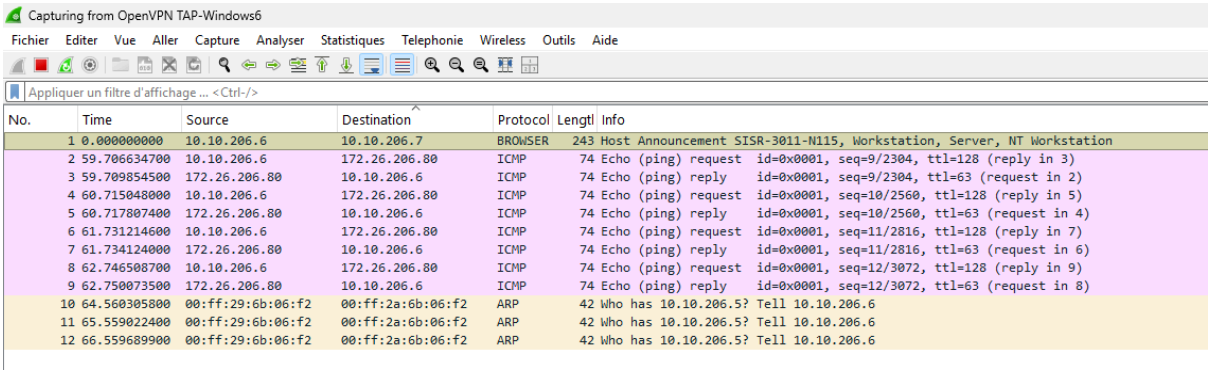




Ici, j'ai lancé une capture sur mon interface Ethernet 3 (ma carte réseau réelle).

J'ai appliqué un filtre sur l'adresse IP publique du serveur (`90.65.104.37`).

Je vois passer des paquets UDP sur le port 20694. C'est exactement ce que je voulais voir : c'est le flux "enveloppe" du VPN. Tout le trafic est chiffré à l'intérieur de ces paquets, donc on ne peut pas voir le contenu, mais cela me confirme que la liaison entre mon PC et le pfSense est bien établie sur le réseau public.



Pour cette capture, j'ai changé d'interface pour sniffer sur la carte virtuelle TAP-Windows6. C'est ici que l'on voit le trafic "décapsulé", tel qu'il ressort du tunnel.

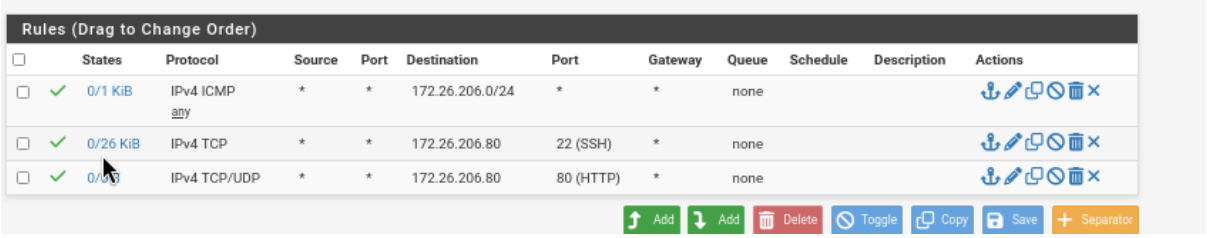
J'ai effectué des pings vers mon serveur interne.

Je vois passer les ICMP Echo (ping) request qui partent de mon IP VPN (`10.10.206.6`) vers le serveur cible (`172.26.206.80`).

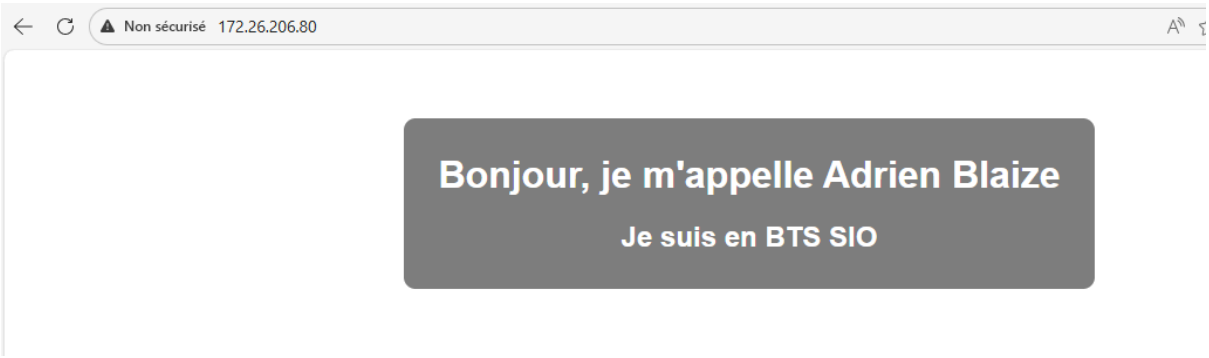
Je vois surtout les ICMP Echo (ping) reply revenir en sens inverse.

C'est la preuve que mon routage est bon et que les règles de pare-feu que j'ai créées sur pfSense (celles autorisant l'ICMP sur l'interface OpenVPN) fonctionnent parfaitement. J'aperçois aussi un peu de trafic ARP et de découverte réseau (BROWSER), ce qui est normal sur un réseau local.

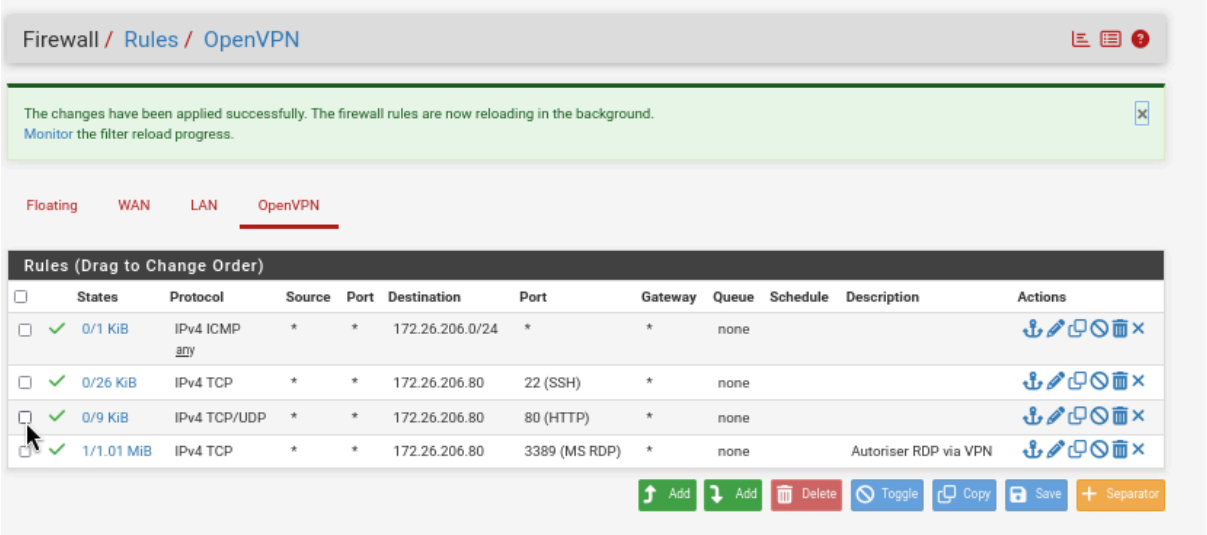
```
333 12.963255300  172.30.11.115  90.65.104.37    UDP  126 49728 → 20694 Len=84
334 12.967266000  90.65.104.37   172.30.11.115   UDP  126 20694 → 49728 Len=84
346 13.967430100  172.30.11.115  90.65.104.37    UDP  126 49728 → 20694 Len=84
347 13.972415600  90.65.104.37   172.30.11.115   UDP  126 20694 → 49728 Len=84
348 14.986182300  172.30.11.115  90.65.104.37    UDP  126 49728 → 20694 Len=84
349 14.989555600  90.65.104.37   172.30.11.115   UDP  126 20694 → 49728 Len=84
407 16.001345900  172.30.11.115  90.65.104.37    UDP  126 49728 → 20694 Len=84
408 16.005071200  90.65.104.37   172.30.11.115   UDP  126 20694 → 49728 Len=84
```



Je suis revenu dans l'onglet des règles de l'interface OpenVPN pour ajouter une autorisation cruciale.
Nouvelle règle : J'ai créé une règle pour autoriser le trafic TCP/UDP sur le port 80 (HTTP) vers mon serveur 172.26.206.80.
Vérification des flux : Je remarque que la colonne States (États) affiche maintenant du trafic (notamment 26 KiB sur la règle SSH). Cela me confirme que mes règles ne sont pas seulement configurées, mais qu'elles sont activement utilisées par ma session VPN.



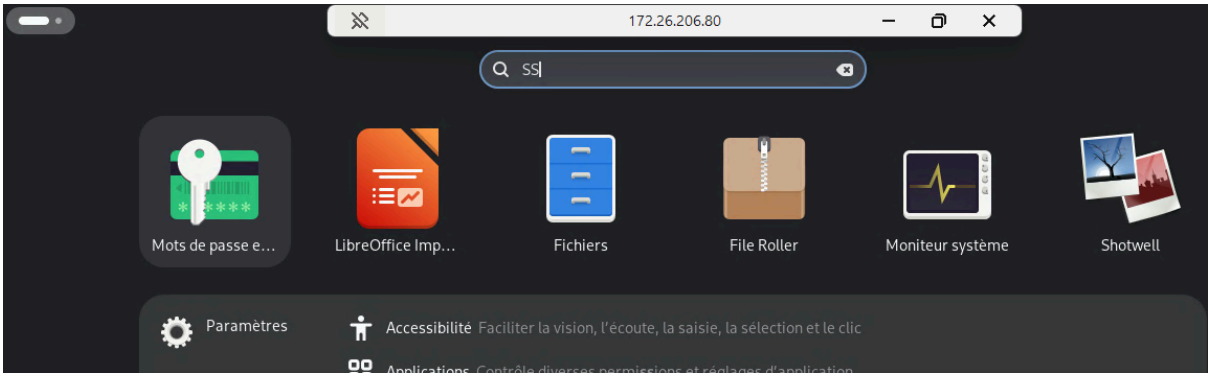
La page s'affiche correctement avec le message « Bonjour, je m'appelle Adrien Blaize / Je suis en BTS SIO ».
Validation : Comme cette adresse IP (172.26.206.80) n'est pas accessible sur Internet, le fait que je puisse voir cette page prouve que mon tunnel VPN fonctionne de bout en bout : le routage est bon, l'authentification a réussi et le pare-feu laisse passer mon flux HTTP.



Ajout de la règle RDP : J'ai créé une quatrième règle pour autoriser le trafic TCP sur le port 3389 (MS RDP) vers mon serveur 172.26.206.80. J'ai ajouté une description (« Autoriser RDP via VPN ») pour faciliter la lecture de ma configuration.
J'ai appliqué les changements, comme le confirme le bandeau vert en haut de l'écran. Le moteur de filtrage a bien rechargé les règles en arrière-plan.
Je constate que cette nouvelle règle est déjà opérationnelle. La colonne States indique 1.01 MiB de trafic, ce qui prouve que je suis actuellement en train d'utiliser une session de bureau à distance à travers le tunnel VPN.

```
root@ablai-Apache:~# apt install xrdp
Installation de :
  xrdp

Installation de dépendances :
  libfuse2t64  xorgxrdp

Paquets suggérés :
  guacamole

Sommaire :
  Mise à niveau de : 0. Installation de : 3Supprimé : 0. Non mis à jour : 241
Taille du téléchargement : 806 kB
  Espace nécessaire : 5 213 kB / 24,9 GB disponible

Continuer ? [O/n] O
Réception de : 1 http://deb.debian.org/debian trixie/main amd64 libfuse2t64 amd64 2.9.9-9 [121 kB]
Réception de : 2 http://security.debian.org/debian-security trixie-security/main amd64 xrdp amd64 0.10.1-3.1+deb13u1 [619 kB]
Réception de : 3 http://deb.debian.org/debian trixie/main amd64 xorgxrdp amd64 1:0.10.2-1 [65,8 kB]
806 ko réceptionnés en 0s (14,2 Mo/s)
Sélection du paquet libfuse2t64:amd64 précédemment désélectionné.
(Lecture de la base de données... 137925 fichiers et répertoires déjà installés.)
Préparation du dépaquetage de .../libfuse2t64_2.9.9-9_amd64.deb ...
Ajout de « détournement de /lib/x86_64-linux-gnu/libfuse.so.2 en /lib/x86_64-linux-gnu/libfuse.so.2.usr-is-merged par libfuse2t64 »
Ajout de « détournement de /lib/x86_64-linux-gnu/libfuse.so.2.9.9 en /lib/x86_64-linux-gnu/libfuse.so.2.9.9.usr-is-merged par libfuse
```

xrdp est un serveur libre et open source qui permet d'implémenter le protocole RDP (Remote Desktop Protocol) de Microsoft sur un système Linux.



Preuve d'une connexion en bureau à distance sur la machine Apache et son bon fonctionnement.