Tutoriel de sécurisation Jellyfin avec Cloudflare Zero Trust

Auteur : Victor Jourdan

Date : 17/03/2026

Table des matières

1) Contexte et objectifs du projet

Au départ, l'accès distant était assuré via OpenVPN sur un NAS Synology DS216play. Cette approche était sécurisée mais limitait fortement les performances a environ 10 Mbps, très loin de la capacite fibre disponible.

L'objectif est donc double : obtenir une diffusion multimédia performante et conserver un niveau de sécurité élevé malgré une exposition sur Internet.

Le principe retenu est une défense en profondeur : reverse proxy local, protection Cloudflare en frontal, politiques Zero Trust, filtrage applicatif, inspection et supervision continue.

2) Prérequis techniques et choix d'architecture

Matériel type : un serveur principal (ex : HPE ProLiant avec GPU Nvidia P4 pour le transcodage) et un NAS secondaire (ex : Ugreen DXP 2800) avec stockage résilient en RAID.

Jellyfin est déployée en conteneur Docker pour simplifier les mises à jour, isoler le service et faciliter la reprise.

Un domaine public est géré chez un registrar (ex : OVH), puis délègue a Cloudflare pour bénéficier du proxy inverse global et des fonctions de sécurité avancées.

Pourquoi ce choix : Docker limite les dépendances système ; Nginx centralise les flux ; Cloudflare absorbe une partie des attaques avant l'infrastructure locale.

3) Déploiement de Jellyfin (Docker) et base réseau

Exemple de déploiement Docker : exposer uniquement le port nécessaire, monter des volumes dédies pour la configuration et les médias, et activer un redémarrage automatique.

Configuration recommandée : réseau dédie au service multimédia, compte de service sans privilèges excessifs et segmentation logique des accès.

Pourquoi : réduire l'impact d'une compromission. Si un composant est touche, l'attaquant ne doit pas pouvoir se déplacer librement vers le reste du SI.

Commande type :

docker run -d --name Jelly fin --restart=unless-stopped -p 8096:8096 -v /srv/jellyfin/config:/config -v /srv/media:/media jellyfin/jellyfin

4) Mise en place du reverse proxy Nginx

Nginx recoit les requêtes externes et les relaie vers Jellyfin en interne. Cette couche permet d'imposer des contrôles supplémentaires (headers, timeouts, limites de débit, journalisation).

Le reverse proxy doit être l'unique point d'entrée depuis Internet vers l'application. Jellyfin ne doit pas être expose directement.

Pourquoi c est critique : on réduit la surface d'attaque et on concentre les contrôles de sécurité sur un composant maitrise et auditable.

Example de directives utile: proxy_set_header X-Forwarded-For, proxy_set_header X-Forwarded-Proto https, client_max_body_size, et timeouts stricts.

5) Redirection NAT/PAT et réduction de surface d'attaque

Configurer la box/routeur pour rediriger le port 443 WAN vers Nginx en LAN. Eviter les ouvertures de ports inutiles.

Pourquoi : chaque port ouvert est un point d'entrée potentiel. Limiter l'exposition au strict minimum est une règle fondamentale.

Recommandation : réserver une IP locale fixe au proxy, documenter les règles NAT/PAT, et vérifier périodiquement que seules les règles nécessaires existent.

6) Configuration DNS et proxy Cloudflare

Creer les enregistrements DNS (A/AAAA) et activer le mode proxy Cloudflare (nuage orange).

Pourquoi activer le proxy Cloudflare : l'IP réelle de l'infrastructure est masquée, ce qui diminue le risque de scan direct et de ciblage de l'origine.

Le réseau Anycast Cloudflare améliore aussi la disponibilité et la latence en rapprochant l'entrée utilisateur du point de présence le plus proche.

7) Configuration SSL/TLS : pourquoi chaque option est activée

Mode SSL/TLS recommande : Full (strict).

Pourquoi Full (strict) : le chiffrement est imposé entre client <-> Cloudflare et Cloudflare <-> origine avec vérification du certificat. Cela évite les intermédiaires non fiables et les attaques de type MITM.

Universal SSL : active automatiquement un certificat Edge pour servir le HTTPS cote visiteur.

Automatico HTTPS Rewrites : corrige certains liens HTTP vers HTTPS pour éviter le contenu mixte et maintenir le chiffrement de bout en bout.

Opportuniste Encryptions : ajoute une couche de protection supplémentaire sur certains flux compatibles.

Toujours désactiver les modes non stricts quand possible pour ne pas dégrader la confiance cryptographique.

8) Mise en place de Cloudflare Zero Trust (Access)

Zero Trust remplace la logique VPN globale par un contrôle d'accès applicatif fin.

Politique type : autoriser uniquement certains pays (ex : France), puis exiger une authentification avec adresse e-mail préapprouvée, puis code à usage unique (OTP).

Pourquoi cette approche : on vérifie identité + contexte avant d'atteindre le serveur. Une requête non conforme est bloquée en amont.

La liste blanche e-mail limite l'exposition aux comptes explicitement valides. Le filtrage géographique ajoute une défense supplémentaire contre des scans opportunistes.

9) Protection advance: WAF, Bot, DDoS, rate limiting

Activer le WAF Cloudflare pour bloquer des signatures d'attaques web courantes (injection, traversa, requêtes anormales).

Activer la protection Bot pour détecter les comportements automatises malveillants.

Activer la protection DDoS pour absorber les pics de trafic hostiles.

Configurer les rates limite sur les Endpoint sensibles (authentification, API) pour ralentir les attaques par force brute.

Pourquoi : ces mécanismes compensent les limites d'un mot de passe seul et réduisent drastiquement la probabilité de compromission.

10) Secure Web Gateway et inspection HTTPS

L'inspection HTTPS (dans un cadre maitrise) permet d'analyser le trafic chiffre pour détecter du contenu malveillant.

Activer l'antivirus sur uploadé/download pour limiter les dépôts de fichiers infectes.

Bloquer les fichiers non scan nables et renforcer l'inspection du corps HTTP pour limiter les contournements.

Pourquoi : de nombreuses menaces se cachent dans des flux TLS légitimes en apparence. L'inspection ajoute une visibilité essentielle.

Attention : bien informer les utilisateurs et respecter les contraintes légales et de conformité en vigueur.

11) Journalisation, supervision et réponse a incident

Conserver les logs DNS, HTTP, réseau, évènements bloques, et tentatives d'authentification.

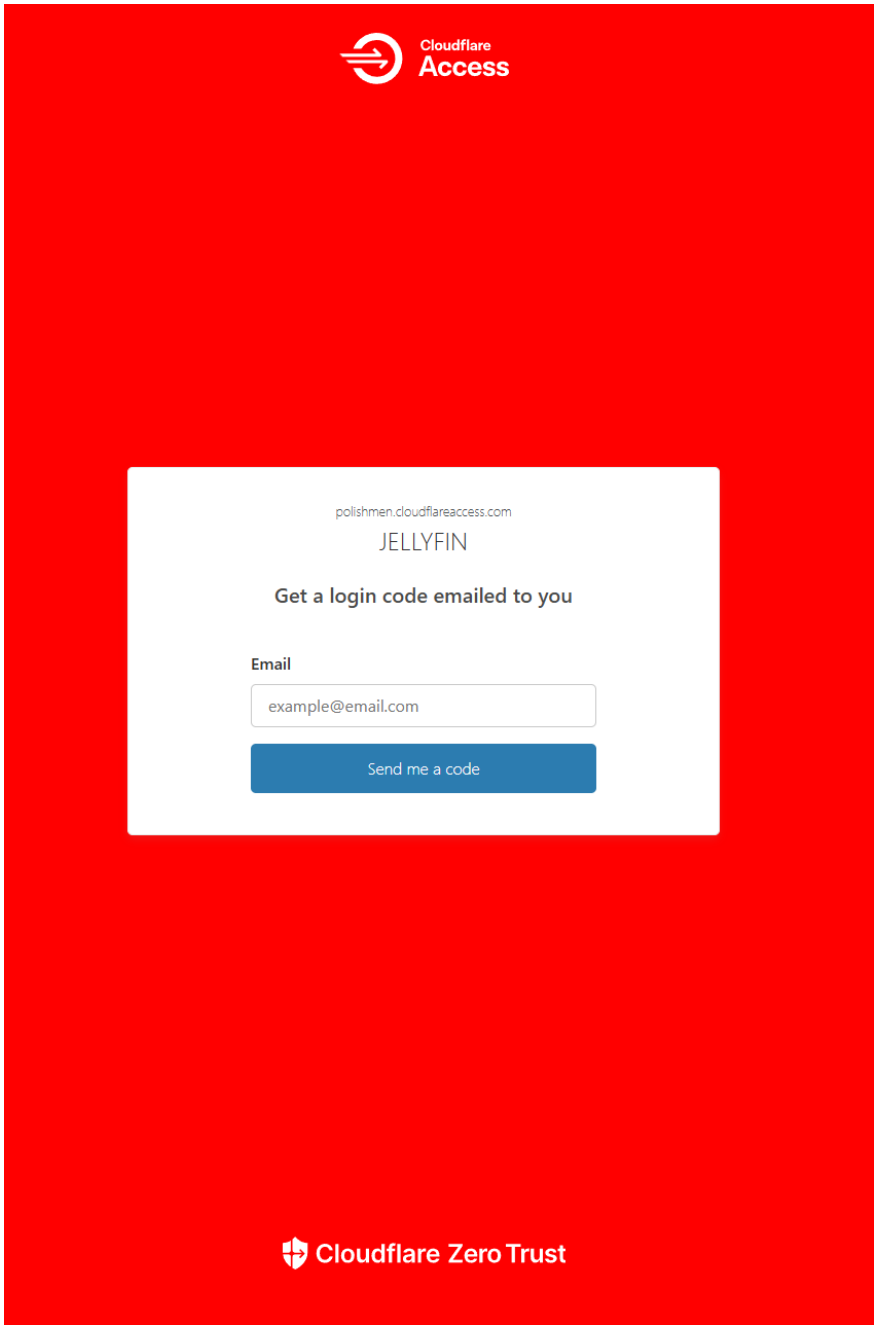
Pourquoi : sans traces exploitables, il est très difficile de comprendre une attaque, corriger rapidement, ou produire un rapport d'incident.

Mettre en place des tableaux de bord : origine géographique du trafic, top URL bloquées, évolutions anormales des bots.

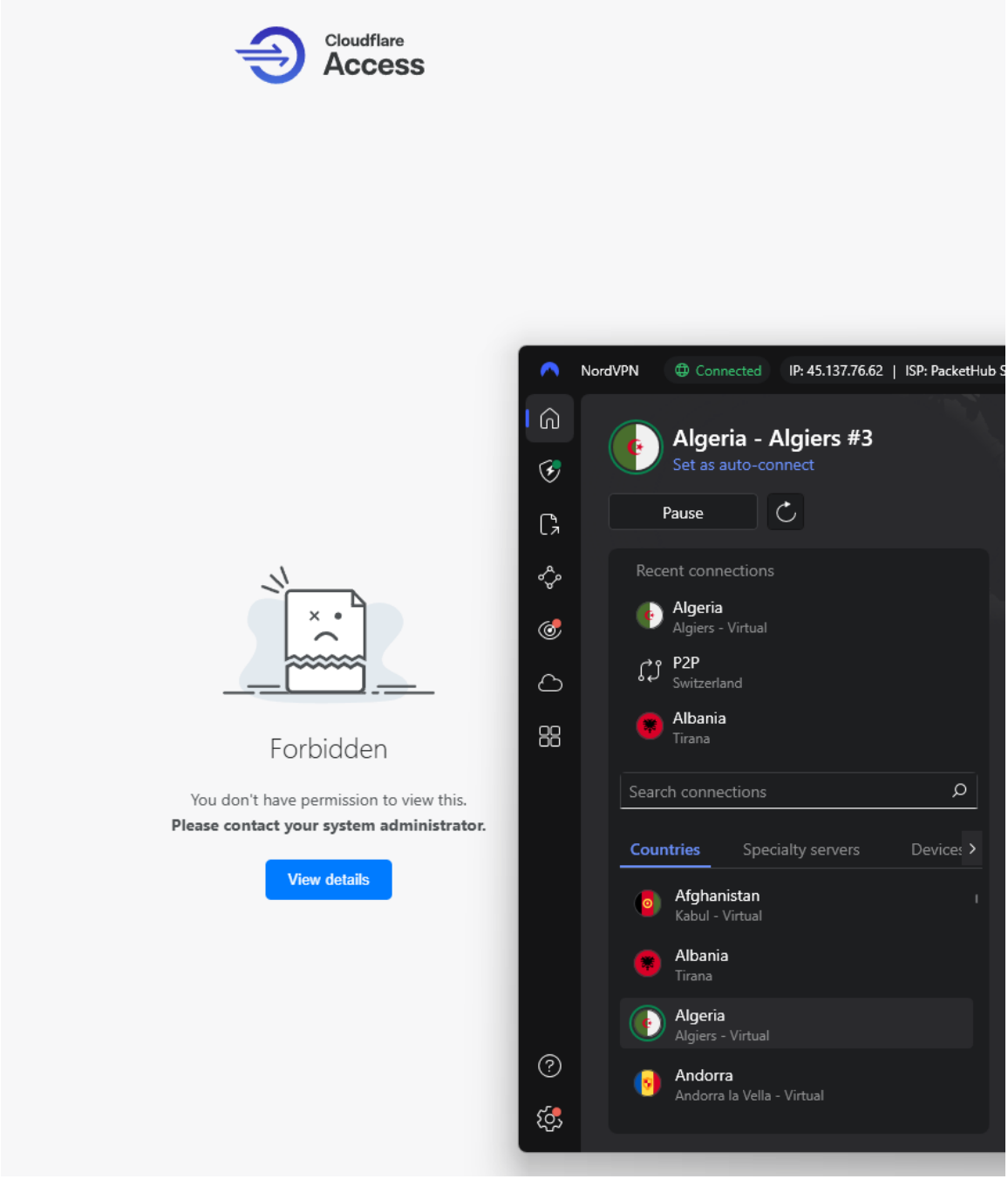
Définir une procédure simple de réponse : détection, contaminent, éradication, retour d'expérience.

12) Scenarios de test et validation de sécurité

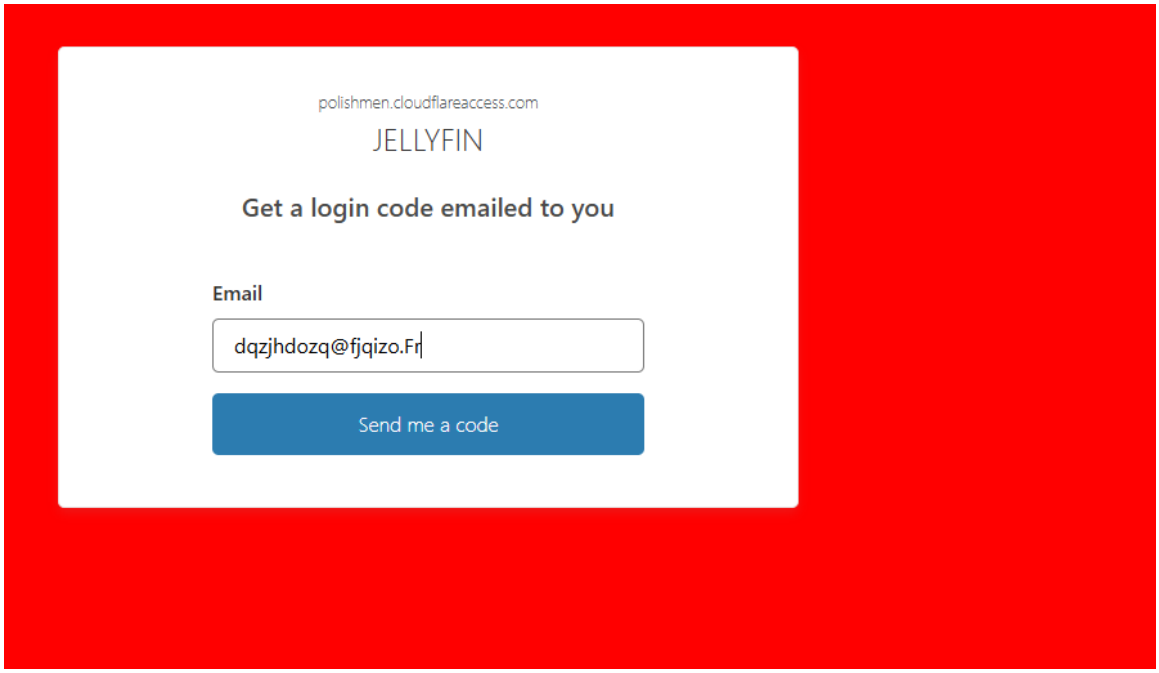
Test 1 : IP française + e-mail autorise -> accès attendu.

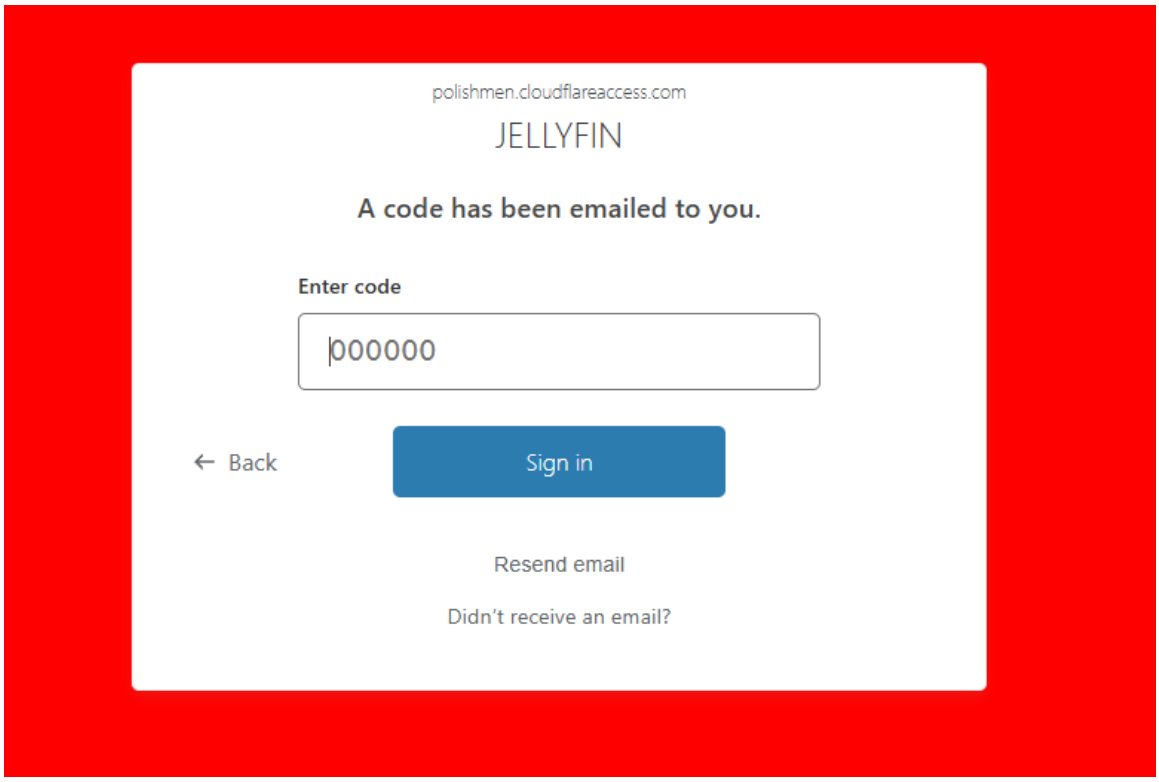


Test 2 : IP étrangère -> blocage attendu avant origine.



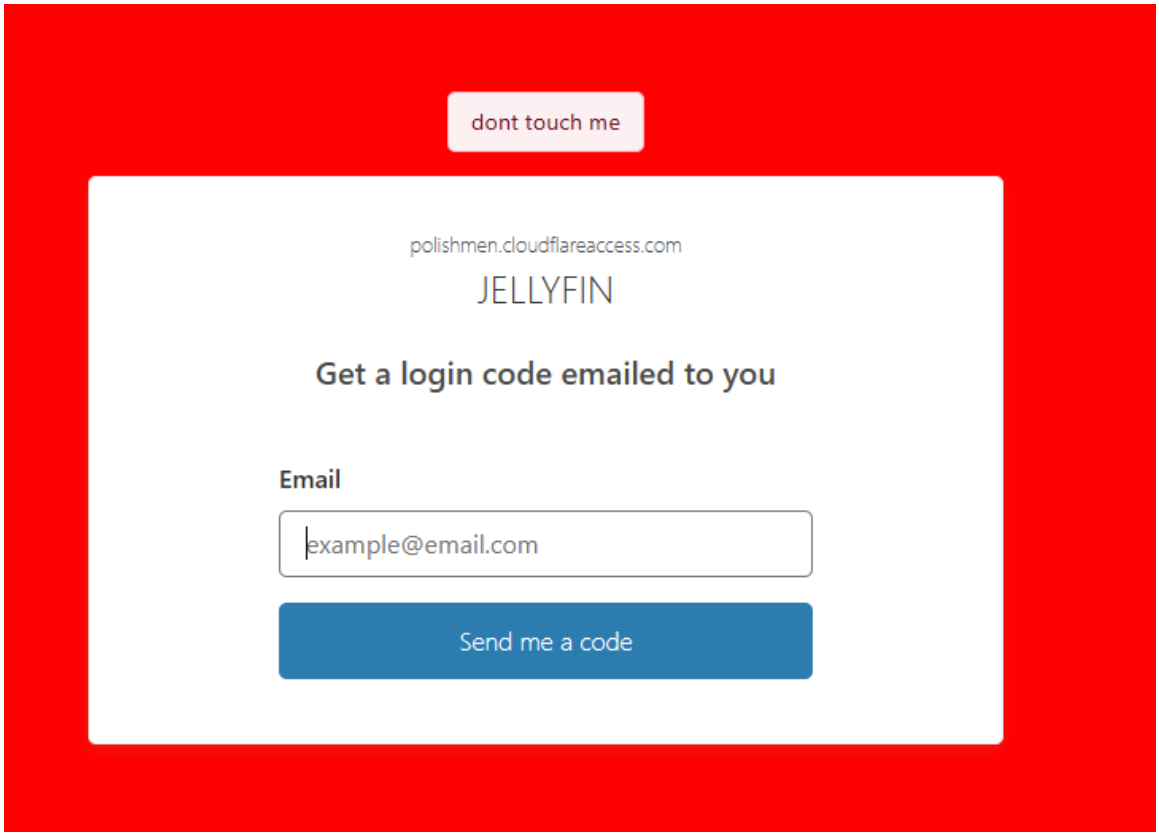
Test 3 : e-mail non whitelist -> blocage OTP/non remise de code.



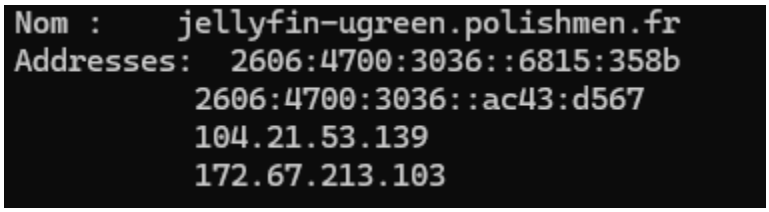


Ici on ne recevra rien naturellement l'adresse mail n'existe pas.

Test 4 : tentatives répétées -> activation de la rate limitent.



Test 5 : vérification du masquage de l'IP origine via outils externes.



Test 6 : test streaming avec transcodage GPU pour confirmer le gain de performance.

Pourquoi ces tests : valider à la fois la sécurité réelle et l'expérience utilisateur.

13) Bonnes pratiques de maintien en condition de sécurité

Mettre à jour régulièrement : Jellyfin, Docker, Nginx, firmware routeur, et politiques Cloudflare.

Faire une revue mensuelle des règles : Access, WAF, rate limites, et journaux.

Activer l'authentification multi facteur pour les comptes d'administration Cloudflare et Jellyfin.

Sauvegarder la configuration et tester la restauration.

Documenter toute modification d'architecture pour conserver une traçabilité complète.

14) Conclusion

La refonte permet de passer d'un accès VPN limite en débit a une architecture Internet performante et sécurisée.

La sécurité repose sur plusieurs couches complémentaires : proxy local, chiffrement strict, Zero Trust, protection applicative et supervision continue.

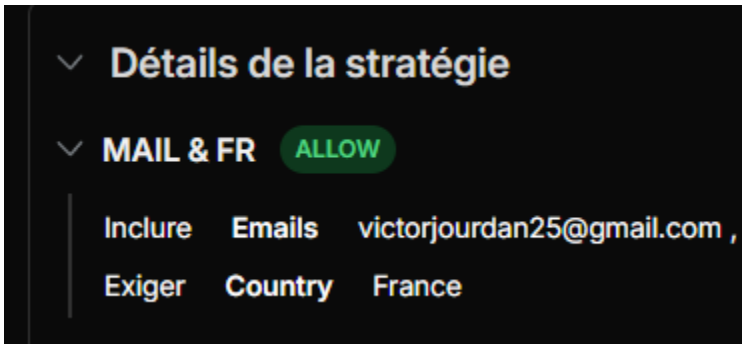
Ce modelé est réutilisable pour d'autres applications métier exposées en ligne, avec les mêmes principes de défense en profondeur.

Annexe :

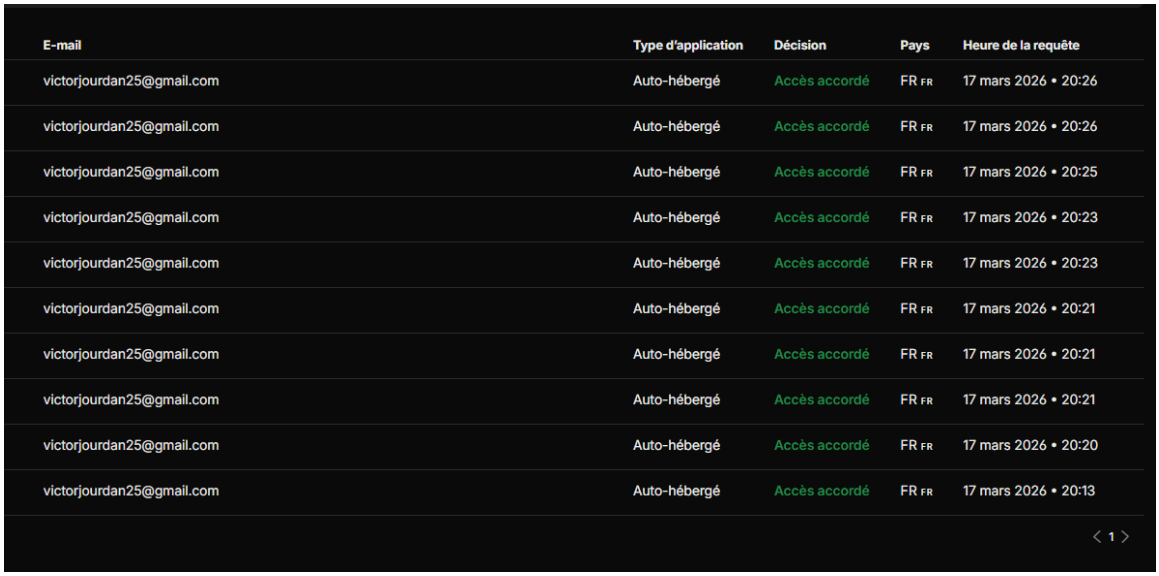
- Dashboard Cloudflare DNS et proxy active

Type	Nom	Contenu	État du proxy	Durée TTL	Actions
A	jellyfin-ugreen	90.65.4.41	Proxied	Automatique	Modifier
A	www	90.65.4.41	Proxied	Automatique	Modifier
CNAME	ftp	polishmen.fr	Proxied	Automatique	Modifier

- Regles Zero Trust Access



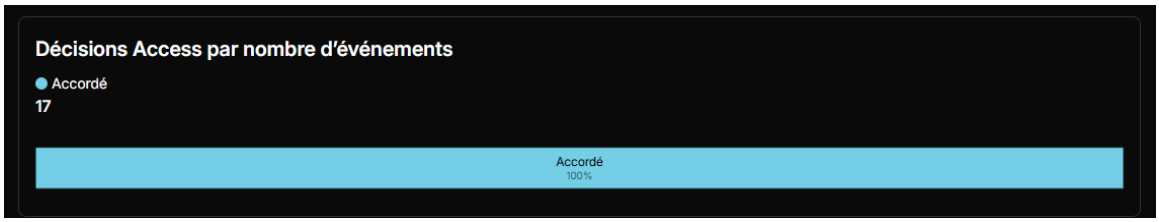
- Exemple de logs de validation

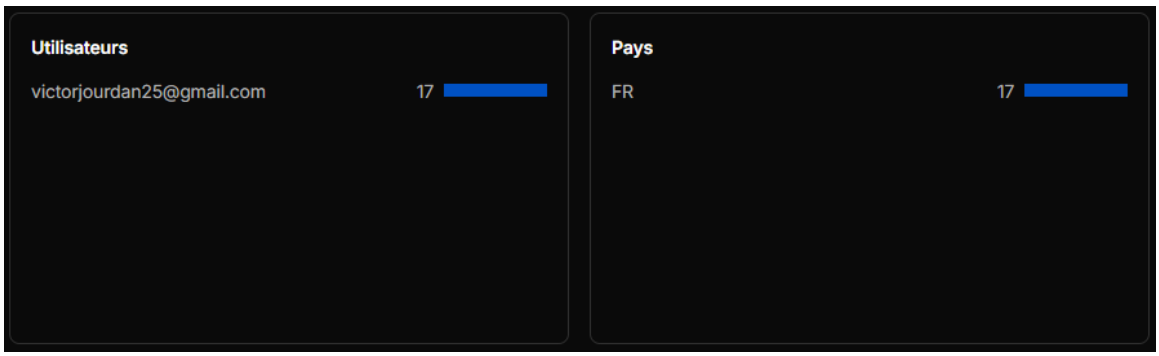


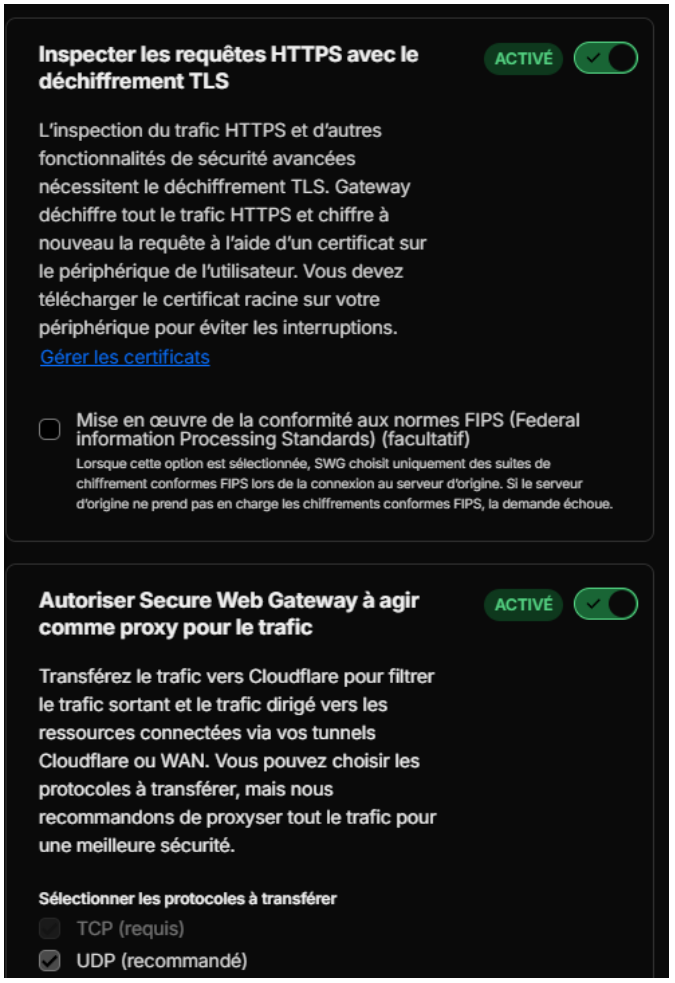
E-mail	Type d'application	Décision	Pays	Heure de la requête
victorjourdan25@gmail.com	Auto-hébergé	Accès accordé	FR FR	17 mars 2026 • 20:26
victorjourdan25@gmail.com	Auto-hébergé	Accès accordé	FR FR	17 mars 2026 • 20:26
victorjourdan25@gmail.com	Auto-hébergé	Accès accordé	FR FR	17 mars 2026 • 20:25
victorjourdan25@gmail.com	Auto-hébergé	Accès accordé	FR FR	17 mars 2026 • 20:23
victorjourdan25@gmail.com	Auto-hébergé	Accès accordé	FR FR	17 mars 2026 • 20:23
victorjourdan25@gmail.com	Auto-hébergé	Accès accordé	FR FR	17 mars 2026 • 20:21
victorjourdan25@gmail.com	Auto-hébergé	Accès accordé	FR FR	17 mars 2026 • 20:21
victorjourdan25@gmail.com	Auto-hébergé	Accès accordé	FR FR	17 mars 2026 • 20:21
victorjourdan25@gmail.com	Auto-hébergé	Accès accordé	FR FR	17 mars 2026 • 20:20
victorjourdan25@gmail.com	Auto-hébergé	Accès accordé	FR FR	17 mars 2026 • 20:13

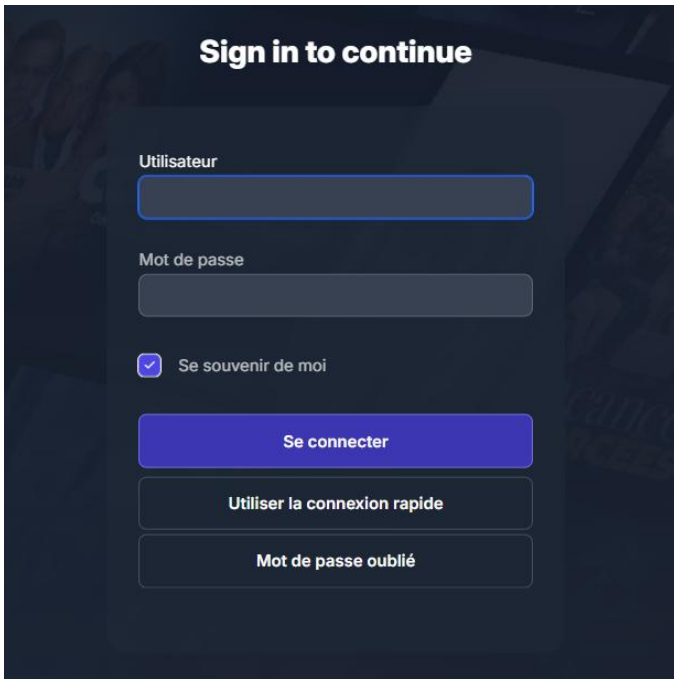
< 1 >

- résultats






Inspecter les requêtes HTTPS avec le déchiffrement TLS
ACTIVÉ
L'inspection du trafic HTTPS et d'autres fonctionnalités de sécurité avancées nécessitent le déchiffrement TLS. Gateway déchiffre tout le trafic HTTPS et chiffre à nouveau la requête à l'aide d'un certificat sur le périphérique de l'utilisateur. Vous devez télécharger le certificat racine sur votre périphérique pour éviter les interruptions.
Gérer les certificats
Mise en œuvre de la conformité aux normes FIPS (Federal information Processing Standards) (facultatif)
Lorsque cette option est sélectionnée, SWG choisit uniquement des suites de chiffrement conformes FIPS lors de la connexion au serveur d'origine. Si le serveur d'origine ne prend pas en charge les chiffrements conformes FIPS, la demande échoue.
Autoriser Secure Web Gateway à agir comme proxy pour le trafic
ACTIVÉ
Transférez le trafic vers Cloudflare pour filtrer le trafic sortant et le trafic dirigé vers les ressources connectées via vos tunnels Cloudflare ou WAN. Vous pouvez choisir les protocoles à transférer, mais nous recommandons de proxyser tout le trafic pour une meilleure sécurité.
Sélectionner les protocoles à transférer
TCP (requis)
UDP (recommandé)


Sign in to continue
Utilisateur
Mot de passe
Se souvenir de moi
Se connecter
Utiliser la connexion rapide
Mot de passe oublié