Sécurisation des accès distants

Procédure de référence. Les adresses, noms d'hôtes et ports de cette page sont des exemples : les transposer à son propre parc.

Le fichier `.md` de cette page : ouvrir le `.md`

Ce que cette procédure couvre : inventorier ce qu'un routeur domestique expose réellement sur Internet, fermer ce qui n'a pas à l'être, durcir ce qui doit rester ouvert, et vivre avec au quotidien.

Le principe directeur : chaque port ouvert est une porte que quelqu'un frappera. La question n'est pas « est-ce risqué ? » mais « ce service a-t-il besoin d'être joignable depuis Internet, ou un tunnel suffit-il ? ». Presque toujours, un tunnel suffit.

> *Convention : chaque bloc indique où il s'exécute — `[routeur]`, `[nas-host]`, `[poste-bureau]`, `[téléphone]`, `[conteneur web]`.*

Le parc de référence

Machine	Adresse	Rôle
Routeur	192.168.0.1	Asus RT-AX86U sous Asuswrt-Merlin
nas-host	192.168.0.11	Hôte Incus + NAS, sans écran ni clavier
Conteneur web `web-srv`	192.168.0.6	ISPConfig + Apache, cible du 443 public
Pi-hole	192.168.0.5	DNS local
Beelink	192.168.0.106	Poste de travail
N73	192.168.0.105	Serveur Minecraft
WireGuard	10.6.0.0/24	Tunnel, serveur en 10.6.0.1

Domaine public : `example.com`. DDNS : `webadmin.asuscomm.com`.

Étape 1 — Inventorier ce qui est réellement exposé

Trois endroits à regarder dans le routeur, pas un seul.

La table de redirections — `[routeur]` → Réseau étendu → Redirection de port. Noter chaque ligne : nom, port externe, port interne, adresse interne, et surtout la colonne IP source, presque toujours vide.

⚠ Le port externe et le port interne peuvent différer. Une règle `8082 → 8080` fait qu'aucune commande lancée sur le serveur ne révélera le 8082 : il n'existe que dans la traduction du routeur.

La DMZ — `[routeur]` → Réseau étendu → DMZ. Un hôte en DMZ reçoit tout le trafic entrant et annulerait tout le reste du travail. Doit être désactivée.

L'UPnP — `[routeur]` → Réseau étendu → Connexion Internet. S'il est actif, n'importe quel logiciel du réseau peut ouvrir un port sans qu'il figure dans la table. L'inventaire est alors incomplet par construction.

Vérifier ensuite ce que le serveur écoute réellement :

```
[nas-host]
incus exec web-srv -- ss -tlnp | grep -E ':808'
```

Resserrer l'UPnP plutôt que le désactiver

⚠ Peu d'applications en dépendent réellement, et il faut vérifier plutôt que supposer. Un jeu en ligne ordinaire — World of Warcraft, ou un client Minecraft rejoignant un serveur — n'ouvre que des connexions sortantes : aucun port entrant, donc aucune demande au routeur. Un serveur Minecraft officiel ne parle pas UPnP non plus, d'où la redirection manuelle qu'il exige. Les outils de visioconférence courants — Teams, Zoom, Meet — n'en dépendent pas davantage :

leur média transite par les serveurs du fournisseur, ou traverse le NAT par STUN/TURN, qui perce un trou depuis l'intérieur au lieu de demander une redirection.

Les vrais utilisateurs sont plus rares qu'on ne croit :

- les clients BitTorrent ;
- les consoles de jeu, qui le réclament pour obtenir un « NAT ouvert » ;
- Syncthing, qui l'active par défaut sur le port 22000 ;
- Plex et Jellyfin, pour leur accès distant ;
- quelques serveurs de jeu auto-hébergés, via des mods.

La table `[routeur]` → Historique du système → Transfert de port tranche la question : elle liste les baux UPnP actifs. Vide alors que les applications tournent, rien n'en dépend — et l'UPnP peut être désactivé sans conséquence.

Si quelque chose en dépend, deux réglages le rendent acceptable sans rien casser :

- « Enable secure UPnP IGD mode » à Oui — une machine ne peut alors demander une redirection que vers elle-même, jamais vers une autre machine du réseau. C'est la protection qui compte le plus.
- Plage de ports externes de 1024 à 65535, et non de 1. Le `1` d'origine permet à une application de réclamer un port privilégié — 22, 443, 3389 —, précisément ceux qu'on vient de fermer.

Les baux existants ne sont pas affectés : la restriction ne s'applique qu'aux nouvelles demandes.

⚠ Vérifier ensuite que les applications qui dépendaient de l'UPnP fonctionnent toujours. Elles échouent en silence : un client BitTorrent privé de port entrant continue de télécharger, simplement moins bien connecté, et n'affiche

rien d’autre qu’un indicateur de couleur.

Cette même page liste la table NAT complète — redirections manuelles et baux UPnP. C’est le seul endroit où l’on voit ce que l’UPnP a réellement ouvert.

□□ Deux lignes `PCREDIRECT` sur le port 80 vers l’adresse du routeur lui-même y figurent normalement : c’est la page que le routeur affiche à la place d’un site quand Internet est tombé. Elle ne s’applique qu’au trafic local — vérifié : le port 80 ne répond pas depuis l’extérieur.

□□ Cas particulier d’une machine derrière un VPN commercial. Si l’application est liée à l’interface du tunnel, sa demande UPnP part vers le VPN et n’atteint jamais le routeur : aucun bail n’apparaîtra, et c’est normal. Le port entrant se demande alors au fournisseur du VPN — Proton le fait par NAT-PMP sur ses serveurs P2P, avec un forfait payant, et son application maintient elle-même la redirection. Il ne reste qu’à donner ce port à l’application et à décocher son propre UPnP, pour qu’un seul acteur tienne le mappage. ⚠□ Ce port change à chaque reconnexion.

Étape 2 — Décider quoi garder

Un seul critère : ce service doit-il être joignable par quelqu’un d’autre que moi ?

Réponse	Décision
Oui — sites web publics, serveur de jeu partagé	Garder
Non, mais j’y accède de l’extérieur	Tunnel VPN, pas de redirection
Non, et je n’y accède que de chez moi	Fermer

Décisions prises ici :

Service	Sort	Motif
HTTPS 443 → .6	Garder	Sert les sites publics
Minecraft 25565 → .105	Garder	Joueurs extérieurs

Service	Sort	Motif
SSH 22 → nas-host	Garder, durci	Usage quotidien SolidExplorer ; Android n’autorise qu’un seul VPN à la fois
FTP 20,21 → .6	Fermer	Non chiffré ; le tunnel donne le même accès
FTP TLS 990 → .6	Fermer	Le tunnel suffit
SSH 8080 → .6:22	Fermer	Tunnel
SSH 8081 → .106:22	Fermer	Tunnel
Panneau ISPConfig 8082 → .6:8080	Fermer	Administration par tunnel
RDP 3389 → .105	Fermer	Premier vecteur des rançongiciels

⚠ Le RDP exposé est le point le plus grave qu’on puisse trouver dans une telle table. Le fermer passe avant tout le reste.

Étape 3 — Préparer avant de fermer

⚠ L’erreur classique : fermer d’abord et découvrir ensuite qu’on dépendait de la porte fermée.

Un nom public résolu vers l’IP publique fait sortir la requête même depuis le réseau local : elle atteint le routeur, qui la renvoie à l’intérieur. Fermer la règle coupe donc aussi l’accès local.

Un enregistrement DNS local — `[Pi-hole]` → Local DNS → DNS Records :

```
web-srv.example.com  →  192.168.0.6
```

Une ligne dans `/etc/hosts` sur toute machine dont le résolveur échappe à Pi-hole :

```
[poste-bureau]
echo "192.168.0.6 web-srv.example.com" | sudo tee -a /etc/hosts
getent hosts web-srv.example.com
```

⚠ Le contournement local d’un VPN commercial couvre le trafic, pas la résolution de noms. Sur le poste-bureau, Proton VPN laisse passer les adresses locales mais envoie les *noms* à son propre résolveur, qui répond l’adresse publique. `/etc/hosts` est consulté avant tout résolveur, VPN ou pas — c’est le seul correctif fiable.

Conséquence à connaître : tant qu’un VPN commercial gère la résolution d’une machine, Pi-hole ne filtre rien de ce que fait cette machine.

Éprouver le tunnel — `[téléphone]`, données cellulaires, WireGuard activé, en visant l’adresse et non le nom :

```
https://192.168.0.6:8080
```

Le nom ne fonctionnera que si le VPN pousse Pi-hole comme résolveur. L’adresse, elle, ne dépend de rien.

Étape 4 — Fermer

`[routeur]` → supprimer les règles décidées, puis Appliquer.

Noter les valeurs avant suppression de toute règle susceptible d’être rétablie :

```
Nom : RDP    Externe : 3389    Interne : 3389    IP : 192.168.0.105    TCP
```

La table de l’Asus n’offre pas d’interrupteur par règle — seulement Éditer et Supprimer.

Se faire depuis le réseau local : aucun risque de se couper l’accès au routeur.

Étape 5 — Vérifier depuis l’extérieur

⚠️ Un test fait avec le VPN actif ne prouve rien : il passe par le tunnel, pas par la porte qu’on vient de fermer.

`[téléphone]`, données cellulaires, VPN coupé :

- l’adresse fermée ne doit plus rien donner ;
- les sites publics doivent continuer de répondre.

ℹ️ Un nom sans site dédié servi sur le 443 affiche le vhost par défaut d’Apache — chez nous, le blog. C’est normal et sans fuite.

Étape 6 — Durcir le SSH qui reste ouvert

Relever l’état effectif

```
[nas-host]
sudo sshd -T \
  | grep -E "^(port|permitrootlogin|passwordauthentication)"
```

`sshd -T` affiche la configuration effective — défauts et fichiers inclus compris — plutôt que ce qui est écrit dans `sshd_config`, qui peut être trompeur.

Le filet — accepter les mots de passe seulement de l’intérieur

⚠️ Ne jamais interdire les mots de passe avant qu’une clé fonctionne. Sur une

machine sans écran, l’erreur oblige à y brancher un clavier et un moniteur.

La forme ci-dessous supprime le risque au lieu de le gérer : elle ferme Internet et laisse le réseau local ouvert.

```
[nas-host]
sudo nano /etc/ssh/sshd_config.d/10-durcissement.conf
```

```
PasswordAuthentication no

Match Address 192.168.0.0/24,10.6.0.0/24
    PasswordAuthentication yes

Match all
```

Deux subtilités de `sshd_config`, inverses des habitudes d’Apache :

⚠ La première occurrence d’une directive l’emporte, pas la dernière. L’`Include /etc/ssh/sshd_config.d/*.conf` se trouvant en ligne 13 et le `PasswordAuthentication` d’origine en ligne 124, un fichier déposé dans `sshd_config.d/` gagne sans qu’on touche au fichier principal — que les mises à jour du paquet peuvent remplacer.

⚠ `Match all` en dernière ligne est indispensable. Un bloc `Match` s’applique à tout ce qui le suit jusqu’au `Match` suivant, et l’inclusion est textuelle : sans lui, le bloc s’étendrait sur tout le reste de `sshd_config`.

Valider avant d’appliquer

```
[nas-host]
sudo sshd -t
sudo sshd -T | grep -i passwordauthentication
sudo sshd -T -C addr=192.168.0.106,user=hostadmin,host=poste-bureau \
  | grep -i passwordauthentication
```

Attendu : silence, `no`, `yes`.

`sshd -T -C` évalue les blocs `Match` pour un contexte donné. C’est le seul moyen de prouver qu’un filet existe *avant* d’en avoir besoin. Aucune de ces commandes n’applique quoi que ce soit.

```
[nas-host]
sudo systemctl reload ssh
```

Ouvrir ensuite une seconde session sans fermer la première : une session établie survit à un rechargement fautif, seule une nouvelle connexion dit la vérité.

Une clé par appareil

Modèle : une clé par appareil, jamais partagée, nommée d’après l’appareil qui la détient. Ce qui doit être conservé n’est pas la clé privée mais `authorized_keys`, qui ne contient que des clés publiques et se sauvegarde sans précaution. Perdre un appareil se règle en retirant sa ligne ; les autres ne sont pas touchés.

```
[poste-bureau]
ssh-keygen -t ed25519 -f ~/.ssh/id_ed25519_pixel9 -C "Pixel9 solidexplorer" -N ""
```

`-N ""` crée la clé sans phrase de passe : un client mobile la redemanderait à chaque connexion et l’outil finirait par être abandonné. Compromis assumé — cette clé vaut par sa révocabilité.

Le carré de caractères affiché par `ssh-keygen` est le randomart, une représentation visuelle de l’empreinte. Purement informatif : rien à copier, rien à noter.

```
[poste-bureau]
ssh-copy-id -i ~/.ssh/id_ed25519_pixel9.pub hostadmin@192.168.0.11
ssh -i ~/.ssh/id_ed25519_pixel9 -o PasswordAuthentication=no \
  hostadmin@192.168.0.11 "echo cle pixel9 OK"
```

⚠️ Le `-o PasswordAuthentication=no` de la seconde ligne est essentiel. Sans lui, une clé défaillante ferait retomber sur le mot de passe, la connexion réussirait, et on croirait la clé valide.

Le poste de travail : phrase de passe et agent SSH

⚠️ Une clé par appareil vaut aussi pour le poste depuis lequel on administre. Créer les clés des appareils mobiles *sur* le poste ne donne aucune clé au poste : il continue de demander le mot de passe à chaque connexion.

Et le compromis s'y inverse. Sur un mobile, une phrase de passe serait redemandée à chaque connexion et l'outil finirait abandonné — d'où le `-N ""`. Sur un poste de travail, l'agent SSH retient la phrase pour toute la durée de la session : on la saisit une fois à l'ouverture, plus jamais ensuite. C'est donc le meilleur des deux, et la phrase se justifie pleinement.

```
[poste]
ssh-keygen -t ed25519 -C "poste vers serveur"
ssh-copy-id utilisateur@192.168.0.11
```

Accepter le chemin par défaut (`~/.ssh/id_ed25519`) : c'est celui que SSH cherche seul, donc rien à préciser ensuite ni à `ssh-copy-id`, ni à la connexion.

À la première connexion, l'invite doit demander `Enter passphrase for key …` et non `…'s password:` — c'est ce qui distingue une clé qui fonctionne d'un mot de passe qui persiste. À la seconde, plus rien n'est demandé.

🔁 Si la phrase est redemandée à chaque connexion, l'agent n'est pas actif dans la session — la plupart des bureaux Linux le lancent automatiquement.

Transférer la clé au client mobile

SolidExplorer accepte le collage direct du texte de la clé, aussi bien qu'un fichier. Le collage évite tout transfert de fichier.

```
[poste-bureau]
cat ~/.ssh/id_ed25519_pixel9
```

Copier tout le bloc, lignes `-----BEGIN` et `-----END` incluses — elles font partie de la clé.

Le déposer dans une note sécurisée du gestionnaire de mots de passe, nommée `Clé SSH privée — Pixel 9 → nas-host`, avec la date et l'utilisateur en commentaire. Le coffre se synchronise vers le téléphone : la note sert à la fois de transfert et de sauvegarde.

`[téléphone]` → SolidExplorer → nouvelle connexion SFTP → « Clé privée SSH » → coller. Utilisateur `hostadmin`.

Proton Drive et Proton Pass sont tous deux chiffrés de bout en bout. La distinction n'est pas la force du chiffrement mais le nombre de copies en clair sur disque : un fichier synchronisé existe en clair sur chaque machine qui le synchronise, et dans les sauvegardes de ces machines. Un secret dans un coffre ne se matérialise nulle part.

⚠ Vérifier qu'aucun outil de synchronisation n'a `~/.ssh` dans son périmètre.

La clé privée peut ensuite être effacée de la machine qui l'a créée, le coffre en étant le dépôt :

```
[poste-bureau]
rm ~/.ssh/id_ed25519_pixel9
```

Une clé par appareil, pas par application

Toutes les applications d'un même appareil partagent la même clé. SolidExplorer et FolderSync sur le Pixel 9 utilisent tous deux `id_ed25519_pixel9` — il n'y a pas de nouvelle paire à générer pour chaque outil. C'est ce qui garde `authorized_keys` lisible et la révocation simple : une ligne retirée bannit l'appareil entier.

Deux formes de transfert selon le client

Les clients mobiles n’acceptent pas la clé de la même façon, et ça change la méthode.

Le client accepte le collage du texte — SolidExplorer, par exemple. C’est le cas simple : la note du gestionnaire de mots de passe suffit, rien ne transite en fichier.

Le client exige un fichier — FolderSync, par exemple. Il faut alors obtenir la forme fichier de la clé. Trois voies, par ordre de préférence :

1. Pièce jointe dans le gestionnaire de mots de passe, si l’offre le permet. Le fichier exact est joint à la note, téléchargé sur l’appareil, et désigné par l’application. Transit et conservation au même endroit, sans risque de déformation.
2. Câble USB entre le poste et l’appareil. Aucun risque de déformation, aucune copie résiduelle.
3. Recréer le fichier sur l’appareil en collant le texte de la note dans un éditeur. ⚠ Certains éditeurs Android ajoutent une extension `.txt` ou convertissent les fins de ligne en CRLF — et une clé en CRLF est refusée sans message clair. À vérifier après coup si la connexion échoue sans raison apparente.

 Ne pas passer par un dossier synchronisé pour ce transfert : la clé privée se retrouverait en clair sur chaque machine synchronisée et dans leurs sauvegardes.

 Si une clé privée revient un jour sur une machine Linux, ses droits doivent être resserrés — SSH refuse d’utiliser une clé lisible par d’autres :

```
[la machine concernée]
chmod 600 ~/.ssh/id_ed25519_pixel9
```

Sur Android la question ne se pose pas : chaque application dispose de son espace propre.

Traduire le port externe

`[routeur]` → modifier la règle SSH : externe 52200, interne 22.

⚠ Ce n’est pas de la sécurité — un balayage complet trouve le port en quelques minutes. Mais les robots ne visent que le 22, et le mot de passe est déjà refusé : le port n’est plus un enjeu de protection, seulement de propreté des journaux. À ce titre, c’est plus efficace que `fail2ban`, qui traite le bruit après coup au lieu de l’éviter.

La modification est entièrement au routeur : le port interne reste 22, la configuration du serveur ne bouge pas, l’accès local est intact.

Coût : un champ à corriger dans le client sur chaque appareil.

Éprouver la chaîne complète

`[téléphone]`, données cellulaires, VPN coupé. C’est le seul test qui exerce la redirection, le refus du mot de passe et l’acceptation de la clé.

Étape 7 — Les interfaces d’administration servies par alias global

Un panneau d’administration peut être public sans figurer dans aucune table de redirections, s’il est servi par le port 443 qu’on a décidé de garder.

Cas rencontré : `https://blog.example.com/phpmyadmin` répondait depuis le réseau cellulaire. Ce n’était pas un lien dans le site, mais un alias Apache global

installé par le paquet, qui répond sous *chaque* vhost.

```
[nas-host]
incus exec web-srv -- grep -Rni "Alias /phpmyadmin" \
  /etc/apache2/ /etc/phpmyadmin/
```

⚠ `grep -r` ne suit pas les liens symboliques rencontrés pendant la descente ; `-R` les suit. Sur Debian, `conf-enabled/` n'est fait que de liens vers `conf-available/`, qui pointe lui-même ailleurs. Une recherche `-r` manque donc le fichier en silence — elle ne renvoie rien, ce qui se lit à tort comme « ça n'existe pas ».

Le correctif : un bloc `<Location>`.

```
[nas-host]
incus exec web-srv -- nano \
  /etc/apache2/conf-available/zzz-phpmyadmin-restrict.conf
```

```
<Location /phpmyadmin>
    Require ip 192.168.0.0/24
    Require ip 10.6.0.0/24
</Location>
```

⚠ `<Location>` l'emporte sur `<Directory>`. Apache applique les sections par type — `<Directory>` d'abord, `<Location>` en dernier — quel que soit l'ordre des fichiers. C'est ce qui permet de passer outre un `Require all granted` posé par un autre paquet sur le même répertoire, sans toucher à un fichier qu'il régénère.

Le préfixe `zzz-` n'a aucun effet technique : c'est un repère de lecture.

```
[nas-host]
incus exec web-srv -- a2enconf zzz-phpmyadmin-restrict
incus exec web-srv -- apache2ctl configtest
```

⚠ `configtest` avant tout rechargement, sans exception : ce serveur porte tous les sites. L'avertissement `AH00548: NameVirtualHost has no effect` est une mention de dépréciation permanente, sans conséquence.

```
[nas-host]
incus exec web-srv -- systemctl reload apache2
```

Ne pas désactiver l'alias avec `a2disconf` : il sert aussi le vhost d'administration interne, et le retirer coupe l'accès de partout, y compris depuis le réseau local.

Vérifier

```
[nas-host]
incus exec web-srv -- curl -sk -o /dev/null \
  -w "%{http_code}\n" -H "Host: blog.example.com" https://127.0.0.1/
```

Origine	Attendu
Machine du LAN	200
Cellulaire avec VPN	200
Cellulaire sans VPN	403
Boucle locale du serveur	403 — normale, hors des plages
Racine des sites	200 — rien n'est cassé

Usage au quotidien

Besoin	Chemin
Panneau ISPConfig, depuis le LAN	`https://web-srv.example.com:8080`
Panneau ISPConfig, de l'extérieur	WireGuard puis `https://192.168.0.6:8080`
phpMyAdmin	`https://blog.example.com/phpmyadmin` — LAN ou VPN
Fichiers sur nas-host, mobile	SolidExplorer, `webadmin.asuscomm.com:52200`, clé
Administration en ligne de commande	WireGuard puis `ssh hostadmin@192.168.0.11`

⚠ Android n’autorise qu’un seul VPN actif à la fois. Un VPN commercial et WireGuard s’excluent sur le même téléphone — c’est ce qui justifie de garder une redirection SSH durcie plutôt que de tout faire passer par le tunnel.

Les pièges, rassemblés

Le port externe peut différer du port interne. Aucune commande lancée sur le serveur ne révélera le port externe.

Un navigateur masque le préfixe `https://` et affiche « Non sécurisé » aussi bien pour du HTTP que pour du HTTPS à certificat accepté manuellement. Ne jamais en conclure qu’un service est en clair : le vérifier.

```
[nas-host]
incus exec web-srv -- curl -sS -o /dev/null \
  -w "%{http_code}\n" http://127.0.0.1:8080/
incus exec web-srv -- curl -skS -o /dev/null \
  -w "%{http_code}\n" https://127.0.0.1:8080/
```

Un `400` en HTTP et un `302` en HTTPS signifient : le port parle TLS, et Apache répond qu’on lui parle en clair.

`PR_CONNECT_RESET_ERROR` sur un port qui devrait servir du HTTPS peut signifier que la requête atterrit ailleurs. Ici, une résolution vers l’IP publique menait au port 8080 externe, redirigé vers SSH : une poignée de main TLS contre un serveur SSH donne exactement ce message.

Un test avec VPN actif ne prouve rien d’un chemin sans VPN.

`grep -r` ne suit pas les liens symboliques ; `-R` oui.

`<Location>` l’emporte sur `<Directory>` dans Apache ; la première occurrence l’emporte dans `sshd_config`. Deux logiques inverses dans deux fichiers voisins.

Vérifier sur quelle machine on tape. Trois machines et un conteneur suffisent à se tromper — une commande sans effet apparent est souvent une commande lancée au mauvais endroit.

Résultat

	Avant	Après
Redirections de port	9	3
Restreintes par IP source	0	—
Mots de passe SSH depuis Internet	acceptés	refusés
Clés SSH sur le parc	aucune	une par appareil
Panneau d’administration public	oui	non
phpMyAdmin public	oui	non
RDP public	oui	non