RomM — Accès distant, publication HTTPS et isolation réseau

Cette page complète les guides **Dolphin** et **Eden**. Elle explique comment rendre le streaming utilisable **en dehors du réseau local**, et comment limiter les risques que cela crée.

> ⚠ **À lire en entier avant de publier quoi que ce soit.** Dans l'état actuel de RomM (5.1.0), rendre le flux accessible depuis Internet revient à exposer un bureau interactif sans authentification. La section 2 détaille exactement ce que cela implique.

---

1. Pourquoi ça ne fonctionne qu'en LAN par défaut

Le bloc `streaming` du `config.yml` contient deux adresses par conteneur, et elles ne sont **pas** utilisées par le même acteur :

```
    - platform: switch
      host: https://192.168.10.100:3713      # ← chargé par le NAVIGATEUR du joueur
      broker_host: http://192.168.10.100:8011 # ← appelé par le BACKEND de RomM
```

- **`broker_host`** est un appel de serveur à serveur. RomM le contacte lui-même, en interne. Une adresse privée convient parfaitement.
- **`host`** est chargé **par le navigateur du joueur**, dans une iframe. RomM ne relaie rien : c'est le poste du joueur qui établit la connexion vidéo directement avec Selkies.

Conséquence : avec une IP privée dans `host`, un joueur hors du réseau obtient un délai d'attente. La page RomM se charge, la session est bien réclamée côté serveur, mais l'image n'arrive jamais.

`host` **doit donc être une adresse joignable depuis le poste du joueur.** Il n'existe que deux façons d'y arriver : amener le joueur dans le réseau (VPN), ou amener le flux jusqu'à lui (publication). Il n'y a pas de troisième voie.

---

2. ⚠ Le trou de sécurité qu'il faut assumer

Le point d'accès du flux n'a aucune authentification. RomM n'attache aucun identifiant à l'URL de l'iframe, et `STREAMING_BROKER_SECRET` — qui existe pourtant — ne protège que le côté `broker_host`, jamais le côté `host`. Il est facile de croire qu'on est couvert alors qu'on ne l'est pas.

Ce que cela signifie concrètement une fois le sous-domaine publié :

- **N'importe qui connaissant l'URL obtient un bureau interactif** dans le conteneur, sans jamais toucher à RomM.
- **L'obscurité ne protège pas.** Chaque certificat Let's Encrypt est publié dans les journaux de *Certificate Transparency*, consultables publiquement. Un sous-domaine « non devinable » y apparaît en quelques minutes. Constaté ici : des robots scannaient le sous-domaine **moins de 24 h** après sa création.
- **La bibliothèque est montée** dans le conteneur. Même en lecture seule, elle reste intégralement copiable.
- `/config` **est en écriture** et contient les clés de console, le firmware et les sauvegardes.
- **Le GPU et la bande passante** sont utilisables par qui obtient le bureau.
- Le streaming **contourne aussi le modèle de permissions de RomM** : un compte en lecture seule peut réclamer une session et se retrouver sur ce bureau. Cela vaut même sur une instance purement locale.

Ce que l'isolation réseau (section 4) apporte — et n'apporte pas

Le cloisonnement décrit plus bas **ferme le pire scénario** : le rebond depuis le conteneur vers le reste du réseau local et vers l'hôte. C'est la mesure la plus rentable, et elle ne coûte rien aux joueurs.

En revanche, elle **ne protège ni la bibliothèque, ni les clés, ni le GPU**. Elle réduit le rayon d'impact, elle ne referme pas la porte.

Vos options, en toute honnêteté

Option	Protection	Friction pour les joueurs
LAN uniquement	Totale	Personne ne joue à distance
VPN (WireGuard, Tailscale)	Totale	Chaque joueur installe un client
Publication (cette page)	Aucune sur le flux	Aucune
Publication + Access List NPM	Réelle	Identifiants à ressaisir à chaque session de navigateur

L'Access List de NPM mérite d'être mentionnée mais déçoit à l'usage : Chrome bloque les invites d'authentification dans une iframe cross-origin, il faut donc visiter le sous-domaine une première fois pour saisir ses identifiants, et la mémorisation ne survit pas à la fermeture du navigateur. Le SSO (Authelia, Authentik…) ne fonctionne pas davantage dans ce contexte, pour les mêmes raisons de cookies `SameSite` et d'en-têtes `X-Frame-Options`.

Une atténuation simple et gratuite

Désactivez les Proxy Hosts entre les sessions. Deux clics dans NPM, et le sous-domaine ne répond plus du tout le reste du temps. La fenêtre d'exposition passe de « permanente » à « quand quelqu'un joue », sans aucune contrainte pour les joueurs.

C'est temporaire

Une PR est en cours côté RomM pour corriger exactement ce point : le broker émettra un jeton de session que RomM ajoutera à l'URL du flux, et un portail le validera avant d'autoriser la connexion.

Suivre rommapp/romm#4173 et l'issue #3968. Cette page devra être révisée à ce moment-là.

---

3. Publier le flux derrière Nginx Proxy Manager

Un sous-domaine par émulateur. Exemple avec `eden.thymon.fr` :

DNS — un enregistrement CNAME `eden` vers votre domaine, exactement comme pour RomM. Vérifier la résolution avant de demander le certificat.

Proxy Host, onglet Details :

Champ	Valeur
Domain Names	`eden.thymon.fr`
Scheme	**https**
Forward Hostname / IP	`eden` (nom du conteneur, voir §4) ou l'IP du serveur
Forward Port	**3001**
Block Common Exploits	activé
Websockets Support	**ACTIVÉ**

> **Visez impérativement le port HTTPS du conteneur (3001), jamais le port HTTP (3000).** Le portail de sécurité installé par le mod n'existe que dans le bloc serveur HTTPS. Proxyfier vers le port 3000 contourne toute protection — l'erreur a été commise ici, et l'interface s'est retrouvée servie sans aucun filtre. NPM ne vérifie pas le certificat en amont, l'auto-signé du conteneur ne pose donc aucun problème.

Onglet SSL : certificat Let's Encrypt, Force SSL, HTTP/2.

Onglet Advanced — le flux est une connexion WebSocket de longue durée, les délais par défaut la couperaient au bout d'une minute d'inactivité :

```
proxy_read_timeout 3600s;
proxy_send_timeout 3600s;
client_max_body_size 0;
```

Puis dans `config.yml`, remplacer uniquement `host:` :

```
      host: https://eden.thymon.fr
```

`broker_host` ne change pas : c'est RomM qui l'appelle, en interne.

> Tant qu'un seul émulateur est migré, la page RomM restera signalée « HTTPS brisé » par Chrome : elle continue de charger du contenu actif depuis les autres origines en certificat auto-signé. L'avertissement ne disparaît qu'une fois **tous** les flux publiés. Si le message persiste ensuite, réinitialisez les autorisations du site dans Chrome — l'option « autoriser le contenu non sécurisé » reste mémorisée.

---

4. Réseau Docker dédié et isolation

C'est la partie qui réduit réellement les risques, et elle apporte aussi de la simplicité.

4.1 Créer le réseau

```
docker network create --subnet 172.30.0.0/24 romm-net
```

Placez-y **RomM et tous les conteneurs d'émulation**, avec une **IP fixe** pour chacun (le champ apparaît dans le template Unraid dès qu'un réseau personnalisé est sélectionné) :

```
romm → 172.30.0.5     eden → 172.30.0.10
dolphin → 172.30.0.11     rpcs3 → 172.30.0.12
```

4.2 Supprimer tous les ports publiés

Une fois les conteneurs sur le même réseau, ils se joignent **par leur nom**. Plus aucun port n'a besoin d'être publié sur l'hôte — ni les ports Selkies, ni surtout **les ports des brokers**, qui commandent le lancement des jeux et n'ont rien à faire sur le LAN.

Dans `config.yml`, utilisez les noms et les **ports internes** :

```
      broker_host: http://eden:8000        # et non http://IP:8011
```

Et dans NPM, `Forward Hostname` = `eden`, port `3001`.

4.3 NPM doit rester sur les deux réseaux

NPM doit rejoindre `romm-net` pour résoudre les noms, **tout en gardant** son accès au LAN pour proxyfier vos autres services. On l'attache donc aux deux :

```
docker network connect romm-net NginxProxyManager
```

⚠ Cette commande **ne survit pas à une recréation du conteneur** (mise à jour, modification de template). Pour la rendre permanente, ajoutez dans le champ **Post Arguments** du template NPM :

```
&& docker network connect romm-net NginxProxyManager
```

Unraid l'ajoute à la fin du `docker run`, donc le rattachement est rejoué automatiquement à chaque recréation — exactement le seul cas où il se perdait.

4.4 Couper les conteneurs du reste du réseau

Deux règles suffisent. Attention : la première ne couvre **pas** l'accès à l'hôte lui-même, car ce trafic passe par la chaîne `INPUT` et non `FORWARD` — d'où la seconde.

Créez un **User Script** planifié sur **At First Array Start** (les règles iptables ne survivent pas à un redémarrage) :

```
#!/bin/bash
SUBNET=172.30.0.0/24
LAN=192.168.10.0/24          # adapter

iptables -D DOCKER-USER -s "$SUBNET" -d "$LAN" -j DROP 2>/dev/null
iptables -I DOCKER-USER -s "$SUBNET" -d "$LAN" -j DROP

iptables -D INPUT -s "$SUBNET" -j DROP 2>/dev/null
iptables -I INPUT -s "$SUBNET" -j DROP
```

Les `-D` en amont évitent d'empiler des doublons si le script est relancé.

Vérification :

```
docker exec eden sh -c 'timeout 3 curl -sI http://IP_UNRAID >/dev/null && echo "PASSE (raté)"
|| echo "BLOQUE (bon)"'
docker exec eden sh -c 'timeout 3 curl -sI http://AUTRE_MACHINE_LAN >/dev/null && echo "PASSE
(raté)" || echo "BLOQUE (bon)"'
```

Les deux doivent afficher **BLOQUE**, et le streaming doit continuer de fonctionner (RomM appelle le broker, jamais l'inverse).

> NPM garde son IP sur le réseau bridge pour joindre le LAN, il n'est donc pas concerné par ces règles. Si vous choisissez de le placer **uniquement** sur `romm-net`, il faudra ajouter des règles `ACCEPT` pour son IP **avant** les DROP.

5. Le portail nginx du mod Eden

Les versions récentes du mod `eden-romm-integration` installent un portail de sécurité : nginx interroge le broker (`auth_request` → `/verify`) et exige un `stream_token` avant de servir quoi que ce soit.

C'est une bonne chose sur le principe — mais RomM 5.1.0 n'envoie pas encore ce jeton. Le résultat est un **403 systématique**, qui se manifeste par un écran noir dans le lecteur. Le journal d'accès du conteneur est sans ambiguïté :

```
GET /?stream_token=<jeton>   → 200, puis GET /websockets → 101   (ouverture manuelle)
GET /                        → 403                               (iframe de RomM)
```

Le mod Dolphin, plus ancien, n'a pas ce portail — d'où une Switch qui ne fonctionne plus alors que la GameCube va bien.

Neutraliser le portail automatiquement

Le portail est réinstallé **à chaque démarrage** du conteneur, donc un `sed` manuel est à refaire sans arrêt. Le point d'accroche `custom-cont-init.d` s'exécute malheureusement **avant** que le mod ne pose le portail : il faut donc y lancer une boucle de surveillance en arrière-plan.

```
mkdir -p /mnt/user/appdata/eden-init
cat > /mnt/user/appdata/eden-init/10-gate-off.sh << 'EOF'
#!/bin/bash
(
  while true; do
    if grep -qE '^[[:space:]]*auth_request /_stream_auth;' /etc/nginx/sites-available/default
2>/dev/null; then
      sed -i '/auth_request off/!s|^\( *auth_request .*\)|#\1|' /etc/nginx/sites-
available/default
      nginx -s reload 2>/dev/null && echo "[gate-off] portail neutralise"
    fi
    sleep 30
  done
) &
EOF
chmod +x /mnt/user/appdata/eden-init/10-gate-off.sh
```

```
chown -R root:root /mnt/user/appdata/eden-init
```

Puis, dans le template Eden, ajouter un chemin :

Champ	Valeur
Container Path	`/custom-cont-init.d`
Host Path	`/mnt/user/appdata/eden-init`
Access Mode	**Read Only**

> Ce dossier est volontairement placé **hors de `/config`** : ces scripts s'exécutent en root, et `/config` est inscriptible par l'application. Sur un conteneur exposé, un répertoire d'exécution root modifiable de l'intérieur serait une porte ouverte. C'est précisément pour cette raison que LinuxServer a sorti ces dossiers de `/config`.

Vérification (compter 40 secondes après le démarrage) :

```
docker logs eden 2>&1 | grep gate-off
docker exec eden grep -n "auth_request /_stream_auth" /etc/nginx/sites-available/default
```

Vous devez voir « portail neutralise » et la ligne commentée.

> **À supprimer le jour où la PR v2 arrive.** Une fois que RomM enverra le jeton, ce portail deviendra une vraie protection — et ce script la désactiverait silencieusement.

---

6. Pièges à connaître

nginx met en cache la résolution des noms. NPM résout `eden` au chargement de sa configuration et garde l'IP. Après une recréation de conteneur, il tape sur l'ancienne adresse → **502 Bad Gateway**, alors qu'un `curl` depuis NPM fonctionne (résolution fraîche). Remède : redémarrer NPM. Les IP fixes évitent le problème.

Toute modification d'un template recrée le conteneur : le portail revient, la couche inscriptible est effacée (paquets installés à la main et journaux nginx perdus), et les patchs du mod sont réappliqués. Un simple redémarrage, lui, conserve tout.

Le journal d'accès nginx est dans le conteneur, à `/var/log/nginx/access.log` — pas sous `/config` . C'est l'outil de diagnostic le plus fiable : il montre l'URI exacte reçue et son code de retour.

Vérifier une exposition depuis l'extérieur : utilisez le journal d'accès ou un téléphone en 4G, en navigation privée. Un navigateur qui possède déjà un cookie de session vous laissera passer alors que tout le monde est bloqué.

---

7. Récapitulatif de l'architecture cible

- Un sous-domaine par émulateur, en HTTPS Let's Encrypt, WebSockets activés, pointant vers le **port 3001** du conteneur
- Tous les conteneurs sur un réseau Docker dédié avec IP fixes, **aucun port publié**
- Brokers joignables uniquement depuis ce réseau, par nom de conteneur
- NPM attaché aux deux réseaux, rattachement rendu permanent par Post Arguments
- Conteneurs coupés du LAN et de l'hôte par deux règles iptables persistées
- Portail Eden neutralisé automatiquement, **en attendant la v2**
- Et l'acceptation assumée que **le flux reste accessible sans authentification** tant que la v2 n'est pas là

Revision #1
Created 2026-08-16 23:15:40 UTC by thymon
Updated 2026-08-16 23:16:02 UTC by thymon