

RAPPORT D'AUDIT DE SÉCURITÉ

Revue complète du code source

AuxiNux Controle Virtua

Hyperviseur web version 1.2.9

Date de l'audit	30 mai 2026
Périmètre	Totalité du code source (~34 400 lignes) : apps/api, apps/runner, apps/vdm, apps/ui, apps/vdm-ui, apps/cli, packages/shared, INSTALL, OS/VirtuaOS
Type	Audit statique manuel (SAST) + analyse des dépendances
Stack	Node.js / TypeScript / Fastify 5 / React 18 / SQLite (better-sqlite3) / argon2
Auditeur	Claude (Anthropic) revue automatisée assistée

Verdict global : la base de code présente une posture de sécurité SOLIDE et mature. Les fondamentaux sont en place (exécution sans shell, requêtes SQL paramétrées, hachage argon2, RBAC par route, validation Zod). Aucune vulnérabilité critique exploitable à distance n'a été identifiée. Les points relevés sont surtout de la défense en profondeur et des durcissements.

1. Synthèse exécutive

Cet audit couvre l'intégralité du dépôt Auxinix Controle Virtua : le backend API (apps/api/src/server.ts, ~7 400 lignes), le service runner exécuté en root (apps/runner), le gestionnaire de datacenter VDM (apps/vdm), les interfaces React (apps/ui, apps/vdm-ui), la CLI, les schémas partagés (packages/shared), ainsi que les scripts d'installation et l'OS VirtuaOS.

Le projet a manifestement déjà fait l'objet de durcissements (audits mars & mai 2026). Les correctifs précédents sont confirmés présents : timingSafeEqual sur le token node-to-node, MAC générées via crypto, verrouillage de compte, politique de mot de passe centralisée, masquage des erreurs 5xx en production.

Décompte des constats

CRITIQUE 0 critique **ÉLEVÉ** 0 élevé **MOYEN** 2 moyens **FAIBLE** 5 faibles
INFO 3 informatifs / observations

Tableau récapitulatif

MOYEN	M-01	Protection CSRF générée mais jamais appliquée (tokens non validés)
MOYEN	M-02	Garde anti-SSRF contournable (liste noire par nom d'hôte + suivi de redirections curl)
FAIBLE	L-01	Identifiants admin par défaut admin/admin123 (API & VDM)
FAIBLE	L-02	API/VDM écoutent sur 0.0.0.0 en HTTP par défaut (trafic en clair sans TLS)
FAIBLE	L-03	Énumération de comptes possible (réponses login 423 vs 401)
FAIBLE	L-04	Injection de configuration LXC via clés extraConfig non filtrées (latent)
FAIBLE	L-05	Opérateurs Docker/VM/LXC H root sur l'hôte (privileged, bind mounts)
INFO	I-01	Runner root accessible via socket Unix confiance par permissions FS
INFO	I-02	Limite d'upload de 8 Go (risque de saturation disque)
INFO	I-03	Tokens d'auth node-to-node stockés en clair en base (compromis LAN)

2. Périmètre & méthodologie

Revue manuelle ligne par ligne des composants sensibles et analyse par motifs sur l'ensemble du dépôt. Axes examinés :

- " Injection (commandes shell, SQL, XML/libvirt, configuration LXC, fstab, iptables) ;
- " Authentification, gestion de session, RBAC et contrôle d'accès par ressource (ACL) ;
- " Protection CSRF, configuration des cookies, en-têtes ;
- " SSRF, traversée de chemin, upload de fichiers, tickets de console WebSocket ;
- " Secrets en dur, génération d'aléa, stockage des mots de passe ;
- " Frontières de privilèges (API runner root VDM nSuds) ;
- " Dépendances (npm audit) et scripts d'installation / VirtuaOS.

Remarque : l'exécution dynamique (DAST) et les tests d'intrusion réseau ne font pas partie de ce périmètre statique.

3. Constats détaillés

MOYEN M-01 Protection CSRF générée mais jamais validée

Composant apps/api/src/server.ts (et apps/vdm/src/server.ts)

Description @fastify/csrf-protection est enregistré (ligne 745) et un jeton est exposé via /api/auth/csrf, mais le plugin n'est jamais branché comme hook/preHandler. Aucune

occurrence de `app.csrfProtection / addHook('preHandler')` n'enforce la validation. Les requêtes mutatives (POST/PUT/DELETE) ne vérifient donc PAS le jeton CSRF. Côté VDM, la route `/join` porte 'config: { csrfProtection: false }' mais aucun hook global n'applique le CSRF non plus.

Impact Théoriquement, une page tierce pourrait déclencher des actions authentifiées. En pratique, le risque est fortement réduit car le cookie de session est en `SameSite=strict`, ce qui empêche son envoi en contexte cross-site sur les navigateurs modernes c'est aujourd'hui la véritable défense CSRF du produit.

Recommandation Brancher réellement la validation : `app.addHook('onRequest', app.csrfProtection)` ou ajouter `csrfProtection` en `preHandler` des routes mutatives, en exemptant `login/csrf`. Conserver `SameSite=strict` comme défense complémentaire. Corriger la note interne qui affirme 'CSRF sur toutes les routes POST'.

MOYEN M-02 Garde anti-SSRF contournable

Composant `apps/api/src/server.ts` : `validateDownloadUrl` (l.1495) ; téléchargement `curl` (l.2897)

Description `validateDownloadUrl` filtre par EXPRESSION sur le nom d'hôte (127., 10., 192.168., 169.254., etc.). Cette approche se contourne par : (1) un nom DNS public résolvant vers une IP privée/mécanismes (DNS rebinding) ; (2) un encodage IP alternatif (décimal `http://2130706433`, hexadécimal, IPv6 mappé [`::ffff:127.0.0.1`]) que `new URL().hostname` ne normalise pas ; (3) surtout, `curl` est invoqué avec `--location` (l.2898) : une URL publique valide peut renvoyer une redirection 302 vers une adresse interne, suivie sans nouvelle validation.

Impact Un opérateur peut faire passer au serveur des requêtes vers des services internes du LAN ou des endpoints de mécanismes cloud (pertinent sur l'hôte OVH/cloud). Privilège requis : compte opérateur authentifié.

Recommandation Résoudre le nom d'hôte et valider l'IP RÉSOLUE (toutes les A/AAAA) contre les plages privées, en bloquant aussi les encodages alternatifs et l'IPv6 mappé. Re-valider chaque saut de redirection (ou désactiver `--location` et gérer les redirections manuellement avec re-validation). Idéalement épingler la connexion à l'IP valide.

FAIBLE L-01 Identifiants administrateur par défaut

Composant `apps/api/src/db.ts` (l.375) ; `apps/vdm/src/server.ts` (l.86) ; `INSTALL/vdm-install.sh`

Description Sans `AUXINUX_INITIAL_ADMIN_PASSWORD`, l'API crée `admin/admin123` avec `must_change_password=1` (rotation forcée bonne atténuation). Le VDM amorce un admin via `BOOTSTRAP_ADMIN_PASSWORD` dont la valeur par défaut est 'admin123', sans rotation forcée équivalente visible.

Impact Fenêtre d'exposition si le service est joignable avant le premier changement de mot de passe. Faible pour l'API (rotation forcée) ; à renforcer côté VDM.

Recommandation Imposer `must_change_password` aussi côté VDM, ou générer un mot de passe aléatoire à l'amorçage. Ne jamais conserver 'admin123' comme valeur par défaut effective en production.

FAIBLE L-02 Écoute sur 0.0.0.0 en HTTP par défaut

Composant `apps/api/src/server.ts` (l.7338, `listen host 0.0.0.0`) ; `cookies` l.738

Description Le serveur écoute sur toutes les interfaces. Le cookie de session n'est 'secure' que si un certificat TLS est chargé (ou `AUXINUX_SECURE_COOKIE=1`). En HTTP pur, identifiants et cookie de session transitent en clair.

Impact Interception possible sur un réseau non fiable. Attenué en usage LAN derrière pare-feu, mais critique si exposé (ex. serveur OVH avec IP publique).

Recommandation Activer TLS dès l'installation (le produit gère ACME/auto-renew), ou restreindre l'écoute à l'interface d'administration. Documenter de ne JAMAIS exposer le panneau en HTTP sur IP publique.

FAIBLE L-03 Énumération de comptes au login

Composant apps/api/src/server.ts : /api/auth/login (I.3106)

Description Un compte verrouillé renvoie 423 (+Retry-After) tandis qu'un identifiant invalide renvoie 401. De plus, argon2.verify n'est exécuté que si l'utilisateur existe, créant un écart de temps de réponse (oracle de présence).

Impact Permet de distinguer les noms d'utilisateur existants. Impact faible (le verrouillage et le rate-limit limitent le bruteforce).

Recommandation Uniformiser le message/codes côté client et exécuter un argon2.verify factice (djà fait dans le VDM avec 'dummy') quand l'utilisateur est absent, pour égaliser le timing.

FAIBLE L-04 Injection de configuration LXC via clés extraConfig

Composant apps/runner/src/handlers/lxc.ts (création, ~I.340)

Description Les VALEURS de config sont protégées (assertSafeConfigValue rejette les retours à la ligne), mais les CLÉS de extraConfig ne sont pas validées. Une clé contenant un saut de ligne pourrait injecter une directive arbitraire (ex. lxc.hook.pre-start) exécution de commande root à l'hôte au démarrage du conteneur. Actuellement NON atteignable : CreateLxcSchema ne déclare pas extraConfig et l'API ne transmet que parsed.data (Zod retire les clés inconnues).

Impact Latent / défense en profondeur. Deviendrait exploitable (par un opérateur) si un futur schéma exposait extraConfig sans valider les clés.

Recommandation Valider les clés au niveau du runner (regex stricte + interdiction des retours à la ligne) au lieu de faire confiance à la couche API.

FAIBLE L-05 Opérateurs H root sur l'hôte (Docker/VM/LXC)

Composant apps/runner/src/handlers/docker.ts (I.393 --privileged ; I.395 bind -v), lxc.ts, qemu.ts

Description La création de conteneurs Docker autorise --privileged et des montages bind d'hôte arbitraires (validés contre la traversée mais pas contre /, /etc ou /var/run/docker.sock). Un opérateur peut donc obtenir un accès root complet à l'hôte. Inhérent à un hyperviseur (comparable à Proxmox).

Impact La frontière de privilège réelle est : tout opérateur autorisé à créer des ressources est de fait root-équivalent sur l'hôte.

Recommandation Documenter clairement ce modèle. Si une séparation est souhaitée, restreindre --privileged et interdire les binds sensibles (/ , /etc, le socket docker) via une liste de refus dédiée, réservée aux administrateurs.

INFO I-01 Runner root via socket Unix

Composant apps/runner/src/runner.ts

Description Le runner s'exécute en root et expose un protocole JSON par lignes sur un socket Unix (/run/auxlinuxvirtual.sock), sans authentification applicative : la sécurité repose sur les permissions du fichier socket (chmod 0o600 dans le code). Tout processus pouvant écrire sur le socket obtient l'exécution root.

Impact Modèle de confiance standard, mais cela implique que le processus API doit partager l'UID/le groupe d'accès au socket donc une compromission de l'API mène directement à l'exécution root via le runner.

Recommandation Confirmer les propriétaire/permissions du socket à l'installation (cohérence avec install.sh). Envisager un durcissement systemd ciblé du runner et, à terme, une allow-list d'actions plus stricte.

INFO I-02 Limite d'upload élevée (8 Go)

Composant apps/api/src/server.ts : fastifyMultipart fileSize 8 Go (I.746)

Description Les uploads (ISO, images) sont plafonnés à 8 Go par fichier. L'gitime pour des ISO, mais

permet de saturer l'espace disque.

Impact D ni de service par remplissage disque possible par un utilisateur autoris .

Recommandation V rifier l'espace libre avant  criture et appliquer des quotas de stockage par utilisateur (la table user_limits existe d j : maxStorageGb).

INFO I-03 Tokens node-to-node stock s en clair

Composant apps/api (datacenter_nodes.auth_token), apps/vdm (vdm_nodes.auth_token)

Description Les jetons d'authentification inter-nSuds sont stock s en clair dans SQLite. Compromis intentionnel d j  document  (usage LAN).

Impact Un acc s en lecture   la base divulgue les jetons inter-nSuds.

Recommandation Acceptable pour un d ploiement LAN ferm . Pour un d ploiement multi-sites, envisager le chiffrement au repos / un secret store.

4. Points forts confirm s

Les contr les suivants sont correctement impl ment s et constituent une base de s curit  solide :

- " Aucune injection de commande : execFile / execFileAsync partout (jamais exec/shell:true). Arguments pass s en tableau, s par s des valeurs utilisateur.
- " Aucune injection SQL : requ tes 100 % param tr es (better-sqlite3 prepare/run/get/all).
- " Mots de passe hach s avec argon2 ; politique centralis e (min 12 car., 1 lettre + 1 chiffre).
- " RBAC appliqu  par route (requireAuth / requireAdmin / requireUiSection) + ACL fine par ressource (requireResourcePermission).
- " Sessions : r g n ration   la connexion, cookies httpOnly + SameSite=strict + secure sous TLS ; saveUninitialized=false.
- " Rate-limit global (1200/min) + login (5/15 min) + verrouillage de compte (10  checs/1h 15 min).
- " Comparaison du token inter-nSuds en temps constant (timingSafeEqual) apr s contr le de longueur.
- " Toutes les routes /api/internal/* exigent le token de nSud (requireInternalNodeToken).
- " Consoles WebSocket prot g es par tickets   usage unique (TTL 60 s), valid s   l'upgrade (ticket + chemin + type).
- " Validation d'entr e robuste : regex ancr es dans le runner + sch mas Zod (safeParse) ; seules les donn es valid es (parsed.data) sont transmises au runner.
- " Travers e de chemin ma tris e : resolvePoolPath (realpath + pr fixe), sanitizeManagedFilename (basename), validatePath/validateDevicePath, r pertoires syst me interdits au montage.
- " Injection de config bloqu e : assertSafeConfigValue (LXC), assertNoFstabMetachars (fstab), regex IP/CIDR + arguments iptables s par s.
- " Al a cryptographique : crypto.randomBytes / randomUUID (MAC, tickets, secrets) plus de Math.random.
- " Gestionnaire d'erreurs masquant les messages 5xx en production (pas de fuite d'info interne).
- " Garde must_change_password appliqu e c   serveur (409 hors routes autoris es).
- " TLS : provisioning ACME + auto-renouvellement + redirection HTTP HTTPS + challenge HTTP-01.

5. D pendances & configuration

npm audit (production) : 0 vuln rabilit . npm audit (incluant dev) : 2 vuln rabilit s mod r es, limit es  esbuild/vite (serveur de dev uniquement, absentes du bundle de production). Conforme   l' tat document  ; Vite reste en v5   cause d'une incompatibilit  @novnc/novnc avec Vite 7.

- " Aucune fuite de secret d tect e dans le code (recherche de motifs)

password/secret/token/clø) : les secrets proviennent de process.env ou sont gønrørs (openssl rand).

- " Le secret de session par døfaut døclenche un avertissement explicite au dømarrage s'il n'est pas surchargø.

6. Plan d'action recommandø

Prioritø 1 (à traiter en premier)

- " Brancher røellement la validation CSRF (M-01) tout en gardant SameSite=strict.
- " Durcir la garde SSRF : validation de l'IP røsolue + re-validation des redirections (M-02).

Prioritø 2

- " Forcer la rotation du mot de passe admin du VDM / Øviter 'admin123' par døfaut (L-01).
- " Imposer/expliciter TLS, ne jamais exposer le panneau en HTTP sur IP publique (L-02).
- " Égaliser les røponses/timing du login (L-03).

Prioritø 3 (døfense en profondeur)

- " Valider les clørs de config LXC cãø runner (L-04).
- " Documenter / encadrer le modLle de privilLges opØrateur (L-05) et durcir le runner (I-01).
- " Quotas de stockage + vØrification d'espace avant upload (I-02).

Conclusion : AuxiNux Controle Virtua est un projet bien construit du point de vue søcuritø, avec des fondations døfensives matures et cohørentes. Les constats sont raisonnables pour un hyperviseur LAN et aucun n'expose de RCE non authentifiøe. Les deux points moyens (CSRF, SSRF) mØritent une correction prioritaire, surtout dLs lors que le serveur est exposø sur une IP publique (cas OVH).

Fin du rapport gønrø le 30 mai 2026.