

EC04 • Rapport de sécurisation • EnerVision

AUTEUR	Johan LEROY, Groupe 3 (HEADL_015B), Tech Lead
DATE DU RELEVÉ	24/09/2026, de 14h45 à 15h15
RÉFÉRENCE DU CODE	commit gelé 9f343e9 du 24/09/2026 à 11h08, porté à la fois par dev et par main
PÉRIMÈTRE	la plateforme EnerVision et sa mise en ligne sur la machine du groupe (conteneur LXC sur l'hôte Proxmox de l'école) : trois environnements Docker Compose (production, recette, dev), chacun derrière son reverse proxy Nginx, et un frontal SNI commun
PREUVES	dossier preuves / , quatorze fichiers numérotés, sorties brutes datées et anonymisées, chacune avec la commande qui la rejoue
FORMAT	source Markdown UTF-8 sans média externe, version figée PDF produite par la chaîne Markdown vers HTML vers CSS de pagination vers PDF

Ce rapport compile des preuves, il ne décrit pas des intentions. Chaque affirmation porte l'un de trois marqueurs :

MARQUEUR	SENS
PROUVÉ	vérifiable dans le dépôt ou par l'API GitHub, rejoué le 24/09 avec sa sortie dans preuves /
CONSTATÉ	relevé sur la machine le 24/09 par une commande en lecture seule (12-constats-machine.txt)
ABSENT	non fait, avec sa raison et son coût

Anonymisation : aucune adresse IP, URL, identifiant de connexion, adresse électronique ni secret n'est reproduit, ici comme dans les preuves. La machine est notée « la machine du groupe », ses noms publics <env>.<domaine-du-groupe> . Le dépôt, lui, a porté l'adresse privée de la machine dans des documents d'exploitation : elle est masquée dans l'arbre livré, mais l'historique git la conserve (section 3).

1. Synthèse

AXE DEMANDÉ	ÉTAT	CE QUI LE PROUVE
Scans laC et code	PROUVÉ onze contrôles rejoués le 24/09 sur le commit gelé, aucune vulnérabilité haute ou critique dans le code ni dans le produit livré ; DAST : 0 échec sur 118 règles	Bandit, pip-audit, npm audit, Checkov, gitleaks, Trivy, terraform validate , nginx -t , OWASP ZAP
Gestion des secrets	PROUVÉ aucun secret dans l'historique git (342 commits hors merges, toutes branches) ; seize secrets générés sur la machine, jamais transmis	gitleaks, git log sur les fichiers sensibles, scripts/provision-host.sh , gardes de config.py

AXE DEMANDÉ	ÉTAT	CE QUI LE PROUVE
Règles réseau	PROUVÉ CONSTATÉ la plateforme n'expose que le frontal SNI, en 80 et 443, tout le reste sur la boucle locale ; CONSTATÉ la machine expose en plus SSH, k3s et trois Vault, hors code livré ; ABSENT pare-feu hôte	ports calculés par <code>docker compose config</code> et <code>provision-host.sh</code> , relevés par <code>ss</code> sur la machine
Authentification	PROUVÉ JWT court, rafraîchissement opaque avec rotation, Argon2id, RBAC à 3 rôles, matrice d'accès testée	86 tests rejoués le 24/09, ADR 0002 et 0003
Audits d'accès et d'infrastructure	PROUVÉ journal d'audit en ajout seul garanti par la base ; historique des déploiements ; supervision Prometheus active en production	ADR 0004 et 0016, state Terraform, API GitHub

Sept constats à traiter, aucun bloquant pour le produit livré, détaillés en section 2.2 :

- La machine expose plus que la plateforme** : SSH en root avec mot de passe, l'API k3s et trois serveurs Vault écoutent sur toutes les interfaces, sans pare-feu hôte. k3s et Vault ne viennent pas du code livré (`12-constats-machine.txt`).
- Aucune approbation humaine avant la production**, contrairement à ce qu'annonce l'ADR 0009, et **aucune branche protégée** (`13-github-reglages.txt`).
- Le chiffrement au repos est impossible sur cette machine** : c'est un conteneur LXC, où LUKS ne peut pas fonctionner. La base est en clair sur le disque ; seules les archives sont chiffrées (SSE-C, ADR 0020).
- Une vulnérabilité MEDIUM** (CVE-2026-41016) dans une dépendance transitive d'Airflow, que **la CI ne peut pas voir** : son audit de dépendances ne porte que sur le verrou du backend.
- Le jeton de réinitialisation de mot de passe passe dans l'URL**, et le journal d'accès du proxy l'enregistre en clair pendant ses quinze minutes de validité (`14-dast-zap.txt`).
- Du code non relu peut tourner sur la machine de production** : toute branche d'un membre, par l'environnement `dev` (ADR 0017), et le workflow qu'apporterait une PR de fork, si son exécution n'est pas soumise à approbation.
- Des fichiers sensibles dans l'arbre de travail du poste** (clé TLS locale, `.env`, state et variables Terraform), ignorés par git mais **qui finiraient dans un ZIP fabriqué à partir du dossier**. Le ZIP du rendu part donc d'un clone.

2. Scans de sécurité

2.1 Résultats

Tous les contrôles ont été lancés le 24/09/2026 entre 14h45 et 15h00 sur `9f343e9`. Checkov, Trivy config et `terraform validate` portent sur un export `git archive` du commit, pour qu'aucun fichier ignoré du poste ne s'y mêle ; Trivy fs porte volontairement sur l'arbre de travail, pour le constat 7.

#	OUTIL	PÉRIMÈTRE	RÉSULTAT	PREUVE
1	Bandit 1.9.4 (SAST Python)	<code>apps/backend/app</code> , <code>ml/enervision_ml</code> , <code>etl/airflow/dags</code>	0 constat, tous niveaux , sur 7 110 lignes (6 136 + 722 + 252)	01
2	pip-audit (verrou figé, sans dev)	backend 50 paquets, ML 94, Airflow 128	backend 0 , ML 0 , Airflow 1 vulnérabilité (PYSEC-2026-24)	02

#	OUTIL	PÉRIMÈTRE	RÉSULTAT	PREUVE
3	<code>npm audit (--package-lock-only)</code>	frontend 511 dépendances, tests e2e 26	0, tous niveaux , dans les deux verrous	03
4	Checkov 3.3.19 (IaC)	Terraform, Dockerfiles, workflows GitHub, secrets	Terraform 0 ressource évaluable · Dockerfile 268 réussis, 3 échecs · workflows 596 réussis , 0 échec · secrets 0	04
5	gitleaks (secrets)	tout l'historique , toutes branches	8 constats, 8 faux positifs après tri ligne à ligne	05
6	Trivy config (IaC)	dépôt entier	3 LOW (HEALTHCHECK), Terraform propre	06
7	Trivy fs (dépendances et secrets)	arbre de travail du poste	la même CVE Airflow, et la clé TLS locale (fichier ignoré par git)	07
8	Tests d'accès du backend	matrice rôle × route, protection des routes, durcissement, JWT	86 réussis , 5 tests d'intégration désélectionnés (joués en CI)	08
9	<code>terraform fmt</code> et <code>validate</code>	les deux racines	formatage conforme, deux configurations valides	09
10	<code>nginx -t</code> et ports fusionnés	proxy de stack, frontal SNI, <code>docker-compose.prod.yml</code>	deux configurations valides , un seul composant exposé	10
11	State Terraform local	racine <code>vm-eni</code>	3 ressources appliquées sur 4 déclarées	11
13	Réglages GitHub	environnements, branches, secrets, runs	voir constat 2	13
14	OWASP ZAP 2.17.0 (DAST)	l'API en fonctionnement, authentifiée	0 échec, 0 avertissement, 118 règles passées , 4 alertes informatives	14

Ce que la CI rejoue, sur chaque PR et chaque push vers `dev` ou `main`, filtré par chemins, derrière le check unique « CI ok » (ADR 0014) : Bandit bloquant à partir de MEDIUM sur le backend et le ML, `pip-audit` sur le verrou du backend, `npm audit --audit-level=high` sur le frontend, `terraform fmt` et `validate`, actionlint et shellcheck sur les workflows, validation des fichiers Compose, `nginx -t` du frontal, `promtool` et `amtool` sur la supervision, une fumée S3 sur Garage avec chiffrement SSE-C, les parcours Playwright et deux tirs k6 (fumée et contrôle du 429 par le proxy), et une analyse SonarCloud. Dependabot suit sept entrées chaque semaine. Le DAST tourne chaque lundi, à la demande, et sur les PR qui modifient son propre workflow. **Ne sont pas en CI** : Checkov, Trivy, gitleaks, `pip-audit` sur les verrous ML et Airflow, `npm audit` sur le verrou des tests e2e. Ils ont été joués pour ce rapport.

2.2 Lecture des constats

1. La machine expose plus que la plateforme. Relevé par `ss` le 24/09 : outre le frontal en 80 et 443, écoutent sur toutes les interfaces SSH (connexion root et mot de passe acceptés), l'API **k3s** en 6443 et **trois serveurs Vault 2.1.1** en 8200 à 8205, initialisés et descellés. Aucun pare-feu ne filtre : la table nftables est vide, en politique `accept`. Ni k3s ni Vault ne viennent du code livré : `git grep vault` ne trouve rien sur le commit gelé. Vault est visé par le Terraform d'une branche de travail non fusionnée. k3s, installé le 17/09, redémarre en boucle depuis (51 678 redémarrages) sans porter aucun pod. Correctif, sans commit : arrêter k3s, restreindre Vault à la boucle locale ou l'arrêter, puis un pare-feu n'ouvrant que 22, 80 et 443, et SSH par clé seule. C'est l'infrastructure de l'administratrice du dépôt : la décision lui revient.

2. Production sans approbation, branches sans protection. L'environnement GitHub `prod` n'a qu'une règle : il n'accepte que la branche `main`. Aucun relecteur n'est requis : le dernier déploiement est passé de l'attente à l'exécution en une seconde. `main` et `dev` ne sont pas protégées, aucun ruleset n'existe : trois commits ont été poussés directement sur `dev` (`c2f360c`, `cbbfaf4`, `6c09bee`), relus ensuite seulement par les PR de remontée #161 et #163. L'ADR 0009 annonçait une production « après approbation » ; l'ADR 0014 en faisait un réglage restant à poser par l'administratrice. Il ne l'a jamais été : les deux ADR portent désormais une note datée du 24/09. Correctif : deux réglages, que seule l'administratrice du dépôt peut activer.

3. Chiffrement au repos. La commande `systemd-detect-virt` répond `lxc` : sans device-mapper ni périphérique loop, LUKS est impossible. `scripts/coffre-luks.sh` le détecte et refuse de démarrer (ADR 0020). La base TimescaleDB et les métadonnées de Garage sont donc en clair sur le disque du conteneur. Ce qui est chiffré dès aujourd'hui : les archives exportées vers Garage par le DAG `retention`, en SSE-C, avec une clé que Garage ne conserve pas. Le chiffrement du disque relève de l'hôte Proxmox, donc de l'administrateur de l'école, à qui la demande est adressée.

4. CVE-2026-41016, MEDIUM, `apache-airflow-providers-smtp 2.3.2`. Le `SmtpHook` d'Airflow négocie STARTTLS sans valider le certificat. Dépendance **transitive** d' `apache-airflow 3.3.2`. **Exposition actuelle : nulle**, aucun DAG n'envoie de courriel et aucune connexion SMTP n'est déclarée dans Airflow. Correctif : `providers-smtp 3.0.0` ou plus. **Le vrai constat est ailleurs** : la CI audite le verrou du backend et pas ceux du ML ni d'Airflow, qui portent 222 paquets, et Dependabot ne suit en `uv` que le backend. La CVE était déjà relevée le 23/09 et elle est toujours là : c'est exactement ce que produit un angle mort.

5. Jeton de réinitialisation dans l'URL (ZAP 10024, informatif). `GET /api/v1/auth/reset-password/validate?token=...` : le jeton est stocké haché, valable quinze minutes, à usage unique, caviardé des journaux de l'API, et `Referrer-Policy: no-referrer` est posé. Mais le journal d'accès du proxy enregistre la requête complète, donc le jeton en clair, lisible par qui administre la machine pendant sa validité. Correctif : passer la validation en `POST`, ou journaliser `$uri` sans ses paramètres sur cette route.

6. Du code non relu peut tourner sur la machine de production. Le troisième environnement, `dev`, se déploie par `workflow_dispatch` depuis n'importe quelle branche (ADR 0017). Il vit sur la même machine et le même démon Docker que la production : tout membre qui a le droit d'écriture peut y exécuter du code non relu. `deploy.yml` n'a jamais de déclencheur `pull_request`, mais cela ne suffit pas, l'ADR 0014 le dit : une PR de fork peut apporter son propre workflow qui cible le runner. La seule protection est alors l'approbation des workflows des contributeurs externes, un réglage que l'API refuse de lire avec les droits d'un membre (403) : non vérifié. Risque accepté pour une équipe de cinq, à fermer avant tout contributeur extérieur.

7. Fichiers sensibles du poste. Trivy fs trouve la clé du certificat auto-signé local, et le poste porte aussi, ignorés : deux `.env`, le state et les variables Terraform, les journaux de session. Tous sont ignorés par git et n'ont jamais été versionnés (05). **Conséquence opérationnelle pour vendredi : le ZIP du dépôt, .git inclus, se fabrique à partir d'un clone**, jamais en compressant le dossier de travail.

HEALTHCHECK absent (Checkov CKV_DOCKER_2, Trivy DS-0026, LOW) sur les images frontend, Airflow et ML. Le backend en porte un, et c'est lui que le proxy attend avant de démarrer ; la base, l'API Airflow et Garage ont le leur dans `docker-compose.yml`. Reste le frontend, un nginx statique qui tomberait sans être signalé. Impact faible.

Terraform : Checkov n'évalue aucune ressource. Les six ressources du dépôt sont des `null_resource` qui agissent par SSH, pour lesquelles Checkov n'a aucune politique. Ce scan ne prouve rien, dans un sens comme dans l'autre. Les garanties réelles sont ailleurs : validation en CI, clé SSH seule, jeton du runner en variable `sensitive` et hors des triggers (11, ADR 0010).

gitleaks : huit faux positifs. Six viennent de la règle `generic-api-key` qui prend `api_history` et `api_current`, deux valeurs de la colonne `source` des relevés, pour des clés. Les deux nouveaux sont des valeurs de test : un identifiant S3 factice (`settings_s3()`) et le mot de passe provisoire simulé d'un test Angular.

3. Gestion des secrets

MESURE	PREUVE	ÉTAT
Aucun secret dans l'historique git, toutes branches ; aucun <code>.env</code> , <code>.pem</code> , <code>.key</code> , <code>.tfvars</code> , <code>.tfstate</code> ni jeton DNS jamais commité	05	PROUVÉ
<code>.env</code> , <code>*.pem</code> , <code>*.tfvars</code> , <code>*.tfstate</code> ignorés par git ; seuls les <code>*.example</code> sont versionnés ; <code>infra/garage/garage.toml</code> , versionné, ne porte aucun secret	<code>.gitignore</code> , 05	PROUVÉ
Seize secrets générés sur la machine par <code>openssl rand</code> (base, API, jeton des métriques, cinq pour Airflow dont sa clé Fernet, six pour Garage dont la clé SSE-C, Grafana, rôle de supervision), <code>.env</code> écrit sous <code>umask 077</code> puis en <code>600</code> . Les secrets existants sont conservés ; le reste du fichier est réécrit à chaque passage pour réaligner hôte, ports et profils	<code>scripts/provision-host.sh</code>	PROUVÉ le script - CONSTATÉ les droits
Refus d'écrire le <code>.env</code> si une valeur d'exemple <code>change_me</code> subsiste, hors identifiants de l'API Mock posés à la main	<code>provision-host.sh</code> , fonction <code>preparer</code>	PROUVÉ
L'API refuse de démarrer si <code>APP_SECRET_KEY</code> fait moins de 32 caractères ou vaut une sentinelle, si <code>APP_DEBUG</code> est vrai hors local, ou si les origines CORS sont en joker ou absentes	<code>apps/backend/app/core/config.py</code>	PROUVÉ
Secrets typés <code>SecretStr</code> , clés S3 et SSE-C comprises, donc jamais journalisés ; jetons, mots de passe et cookies caviardés dans les journaux	<code>config.py</code> , <code>app/core/logging.py</code>	PROUVÉ
Compose exige par <code>\${VAR:?}</code> les secrets de la base et de l'API ; ceux d'Airflow, de Garage et de la supervision sont gardés par <code>airflow-init</code> et par les gardes du <code>Makefile</code> avant tout démarrage	<code>docker-compose.yml</code> , <code>Makefile</code>	PROUVÉ
Clé de signature d'Airflow distincte de celle de l'API	<code>docker-compose.yml</code>	PROUVÉ
Côté GitHub, un seul secret (<code>SONAR_TOKEN</code>) ; le déploiement n'en consomme aucun	13	PROUVÉ
Terraform : clé SSH seule, jeton du runner en variable <code>sensitive</code> , absent des triggers	11, ADR 0010	PROUVÉ
Jeton du scan DAST éphémère et caviardé des journaux publiés	<code>dast.yml</code> , <code>scripts/dast-token.sh</code>	PROUVÉ

Ce qui manque. Le dossier EC01 prévoyait **SOPS + age** : non fait. Les secrets vivent en clair sur le disque de la machine, protégés par les droits du fichier seulement, et la rotation est manuelle. Le 23/09, ce rapport jugeait la situation acceptable pour deux environnements, mais plus pour trois : **le seuil est franchi**, avec trois `.env`, un jeton DNS et une clé SSE-C dont la perte rendrait les archives illisibles. L'ADR 0019 demande de sauvegarder cette clé

hors de la machine : c'est une procédure, rien ne le vérifie. Enfin, l'adresse privée de la machine figure dans l'historique git (ADR d'exploitation du 21 et du 22/09) : adresse non routable, joignable seulement depuis le réseau de l'école, masquée dans l'arbre livré.

4. Règles réseau

4.1 Surface exposée

Sur la machine, `provision-host.sh` ramène tous les ports des trois stacks sur la boucle locale (`10`) :

COMPOSANT	PUBLICATION SUR LA MACHINE	JOIGNABLE DEPUIS
Frontal SNI (<code>infra/front</code> , nginx <code>stream</code>)	réseau de l'hôte, 80 et 443	le réseau
Proxy Nginx de chaque stack	<code>127.0.0.1</code> : 10443, 8443, 9443 (HTTPS) et l'écouteur PROXY protocol du frontal	la machine seule
Base, Mailpit, API Airflow	<code>127.0.0.1</code> , un port par environnement	la machine, donc par tunnel SSH
Garage (S3 et administration)	<code>127.0.0.1</code> , un port par environnement ; pas encore en <code>dev</code> , resté sur le commit du 23/09	la machine seule
Prometheus, Alertmanager, Grafana (production)	<code>127.0.0.1</code>	la machine seule
Backend, frontend, scheduler et processeur Airflow, exporteurs	aucune	le réseau interne de Compose
Hors plateforme : SSH, API k3s, trois Vault	toutes les interfaces : 22, 6443, 8200 à 8205	le réseau (constat 1)

Un seul composant exposé par la plateforme. Le frontal lit le nom demandé (SNI) sans déchiffrer et relaie la connexion, en PROXY protocol, vers le proxy de la stack visée : l'adresse réelle du client arrive jusqu'à la limitation de débit. Les trois environnements sont trois projets Compose distincts, sans réseau ni volume partagé (ADR 0009, 0017, 0018). Sur la machine, `ss` confirme les ports de la plateforme, mais relève aussi les services hors plateforme du constat 1. [Constaté, `12`]

4.2 Reverse proxy et TLS

DIRECTIVE	VALEUR	EFFET
Certificats	Let's Encrypt par défi DNS-01 (<code>acme.sh</code> , domaine <code>dynv6</code>), vérifiés chaque nuit par une tâche cron et renouvelés à échéance ; l'auto-signé ne sert plus qu'au poste et aux tests <code>e2e</code>	chaîne de confiance publique, sans port 80 ouvert pour le défi (ADR 0018)
Protocoles	TLS 1.2 et 1.3, tickets de session désactivés	pas de protocole obsolète
Redirection	80 vers 443	pas de trafic applicatif en clair
<code>Strict-Transport-Security</code>	<code>max-age=31536000; includeSubDomains</code>	le navigateur refuse ensuite le HTTP
<code>Content-Security-Policy</code>	<code>default-src 'self', frame-ancestors 'none' ...</code>	limite l'injection de script et le clickjacking

DIRECTIVE	VALEUR	EFFET
En-têtes de l'API	<code>nosniff</code> , <code>X-Frame-Options: DENY</code> , <code>Referrer-Policy: no-referrer</code> , <code>Cross-Origin-Resource-Policy: same-origin</code>	défense en profondeur si le proxy manquait
Limitation de débit	<code>api</code> 20 req/s, <code>auth</code> 30 req/min sur les routes qui vérifient un secret ; contrôlée par un tir k6 en CI	freine la force brute sans pénaliser la navigation derrière le NAT de l'école
	<code>server_tokens off</code> , corps limité à 2 Mo	pas de version publiée, pas de requête démesurée

Les deux configurations sont validées par `nginx -t` sur l'image de production (10). Certificats servis : Let's Encrypt (émetteur YE1) sur les trois noms, échéance au 22/12/2026, et les trois sondes de santé répondent 200 par le nom public avec un certificat vérifié, sans `-k`. [Constaté, 12]

4.3 Accès à la machine et déploiement

- Le **runner GitHub Actions** auto-hébergé initie lui-même la connexion vers GitHub : **aucun port entrant** n'est ouvert pour déployer. **PROUVÉ**
- `deploy.yml` ne se déclenche **jamais** sur `pull_request` : sur un dépôt public, une PR venue d'un fork exécuterait sinon son code sur la machine. Il n'est appelé qu'après une CI verte sur un push, et refuse de revenir à un commit plus ancien que celui déployé. **PROUVÉ**
- La production n'accepte que `main`, **sans approbation humaine** (constat 2). [Prouvé, 13]
- Terraform se connecte par **clé SSH**, jamais par mot de passe. **PROUVÉ**

Ce qui manque. **Aucun pare-feu hôte** n'est configuré ni documenté : la restriction repose sur la publication des ports et sur le réseau de l'école ; la table nftables est vide, en politique `accept` [Constaté, 12]. **SSH n'est pas durci** : connexion root et authentification par mot de passe acceptées [Constaté, 12] ; Ansible était prévu au dossier EC01 et n'a pas été fait. Pas de réseaux Docker nommés : l'intention du dossier EC01 (la base jamais exposée) est tenue par l'absence de publication, plus fragile à la relecture qu'une politique explicite. Azure n'est pas utilisé, par choix d'architecture on-premise : il n'y a ni NSG ni Application Gateway à auditer.

5. Authentification et autorisation

MÉCANISME	DÉTAIL	TRACE
Jeton d'accès	JWT HS256, 15 minutes , algorithme épinglé, <code>aud</code> , <code>iss</code> et <code>typ</code> vérifiés, <code>alg: none</code> rejeté, gardé en mémoire côté navigateur	ADR 0002, <code>app/core/security.py</code>
Jeton de rafraîchissement	chaîne opaque de 256 bits , stockée hachée, rotation à chaque usage et détection de réutilisation ; cookie <code>HttpOnly</code> , <code>Secure</code> , <code>SameSite=Strict</code> , chemin restreint	ADR 0002, <code>app/services/auth.py</code>
Mots de passe	Argon2id (m=19 456 Kio, t=2, p=1), re-hachage passif si les paramètres changent	<code>app/core/hashing.py</code>
Réinitialisation	jeton haché, 15 minutes, usage unique ; voir le constat 5	<code>app/models/password_reset_token.py</code>

MÉCANISME	DÉTAIL	TRACE
Force brute	limitation à fenêtre glissante sur trois clés, évaluée avant le hachage ; pas de verrouillage de compte, qui serait un déni de service	ADR 0002, <code>login_attempt</code>
Énumération de comptes	message et temps de réponse identiques quelle que soit la cause	<code>app/services/auth.py</code>
Autorisation	RBAC à trois rôles ordonnés (<code>lecteur</code> , <code>opérateur</code> , <code>admin</code>), décision prise sur la ligne en base relue à chaque requête , jamais sur le claim	ADR 0003, <code>app/api/deps.py</code>
Révocation	immédiate : compte désactivé ou mot de passe changé invalide les jetons antérieurs	<code>credentials_changed_at</code>
Refus par défaut	16 routes sous rôle , 3 authentifiées sans rôle, 1 par cookie, 8 publiques listées nommément (dont <code>/metrics</code> , gardée par son propre jeton) ; un test appelle réellement chaque route sans jeton	<code>tests/api/acces.py</code> , <code>test_route_protection.py</code>
Matrice d'accès	chaque route gardée croisée avec les trois rôles, sur les routes réelles, puis rejouée avec de vrais jetons contre une vraie base en CI	<code>test_matrice_acces.py</code>
Garde-fous d'administration	refus de rétrograder ou désactiver le dernier administrateur ; premier administrateur créé hors dépôt	<code>app/services/user.py</code> , <code>app/cli.py</code>

Preuve d'exécution : 86 tests d'accès, de protection des routes, de durcissement et de JWT, réussis le 24/09 (08) ; le DAST authentifié ne relève ni échec ni avertissement (14).

Ce qui reste ouvert, et c'est écrit dans le dépôt (`owasp-traceabilite.md`) : **pas d'autorisation par objet** (OWASP API1). Les rôles sont globaux, un compte `lecteur` lit tous les sites. Sans conséquence tant que les routes métier sont en lecture seule ; à corriger par une table d'affectation compte-site avant la première route d'écriture.

6. Audits d'accès et d'infrastructure

6.1 Traçabilité applicative

- **Journal d'audit en ajout seul, garanti par PostgreSQL** : deux déclencheurs refusent `UPDATE` , `DELETE` et `TRUNCATE` sur `audit_log` . Les champs de détail passent par une liste blanche ; l'acteur est dénormalisé pour survivre à la suppression d'un compte (ADR 0004). **PROUVÉ** par les tests d'intégration • en production, 5 créations de compte et 3 changements de mot de passe [Constaté, 12]
- **Tentatives de connexion** journalisées à part (`login_attempt`). **PROUVÉ**
- **Limite assumée** : les déclencheurs arrêtent l'accident, pas un compte qui détient `ALTER TABLE` . Le journal n'est pas une preuve de non-répudiation.

6.2 Traçabilité de l'infrastructure

- **Provisionnement** : le state Terraform porte trois ressources appliquées (Docker, les trois environnements, le runner). La quatrième, le coffre LUKS, n'a jamais été appliquée (11 , constat 3).

- **Déploiements** : chaque déploiement est un run de `deploy.yml` rattaché à un environnement GitHub. Sept déploiements de production entre le 23/09 11h37 et le 24/09 11h12, le dernier sur le commit gelé (13).
- **Dépôt** : une seule administratrice, aucune branche protégée (constat 2).

6.3 Supervision

PROUVÉ dans le dépôt, active en production (profil Compose `monitoring`, ADR 0016) : Prometheus lit `/metrics` avec son jeton, ainsi que la base, l'hôte, les conteneurs et Garage. Neuf règles d'alerte sont testées par `promtool` en CI : API indisponible, erreurs serveur, latence, base indisponible ou saturée, mémoire, disque et CPU de l'hôte, cible injoignable. Alertmanager les envoie à Mailpit. Trois tableaux Grafana couvrent l'API, les données et l'infrastructure. En production, les sept cibles sont `up` et les neuf règles chargées [Constaté, 12].

Ce qui manque : aucune règle sur des **événements de sécurité** (pic de 401, de 403 ou de 429) ; des alertes qui restent dans Mailpit et ne réveillent personne ; pas de Loki, donc aucune centralisation des journaux ; aucune alerte sur l'échec d'un DAG.

7. Écarts avec le dossier EC01, et leur coût

PRÉVU AU DOSSIER EC01	LIVRÉ	COÛT
Traefik en terminaison TLS	Nginx par stack (ADR 0007), plus un frontal SNI (ADR 0018)	configuration écrite à la main, mais explicite et validée en CI
Trivy, Bandit, gitleaks en CI	Bandit bloquant en CI ; Trivy, gitleaks et Checkov joués pour ce rapport	un secret commité demain ne serait vu qu'au prochain passage manuel
SOPS + age	secrets générés sur la machine, jamais transmis	secrets en clair sur disque, sans sauvegarde vérifiée
Ansible pour le durcissement	rien	pare-feu et SSH non durcis de façon reproductible
Deux réseaux Docker	un seul composant exposé	propriété portée par une absence, fragile à la relecture
Prometheus, Grafana, Loki	Prometheus, Alertmanager, Grafana actifs en production ; pas de Loki	journaux dispersés, aucune alerte de sécurité
Scan d'image de conteneur	aucun	les images construites sur la machine ne sont pas analysées
Chiffrement au repos	SSE-C des archives ; LUKS écrit mais impossible sur LXC (ADR 0020)	base en clair sur le disque du conteneur

8. Plan d'action

8.1 Avant le gel du 25/09, 9h00

1. **Fabriquer le ZIP du dépôt depuis un clone**, `.git` inclus, et vérifier qu'il ne contient ni `.env`, ni `*.pem`, ni `*.tfvars`, ni `*.tfstate`.

- 2. Activer, par l'administratrice : un relecteur requis sur l'environnement `prod` , et la protection de `main` . Deux réglages, sans commit.
- 3. Arrêter k3s, qui redémarre en boucle, et restreindre les trois Vault à la boucle locale (constat 1). Sans commit, sur décision de l'administratrice.

8.2 Après le gel, par ordre de valeur

- 1. Pare-feu hôte n'ouvrant que 22, 80 et 443, et SSH par clé seule, dans un script rejouable.
- 2. `pip-audit` sur les trois verrous, gitleaks et Trivy en CI ; montée de `providers-smtp` .
- 3. Validation du jeton de réinitialisation en `POST` , ou journal d'accès sans paramètres.
- 4. Chiffrement du disque par l'hôte Proxmox, ou une vraie machine virtuelle pour dérouler le coffre LUKS déjà écrit.
- 5. Environnement `dev` sur une autre machine, ou limité aux branches relues.
- 6. Règles d'alerte de sécurité et Loki ; seuil bloquant sur le DAST.
- 7. Autorisation par site (API1) ; SOPS + age ; HEALTHCHECK du frontend.

9. Contribution personnelle

Attribution vérifiée par `git log` sur chaque fichier cité.

SUJET	AUTEUR PRINCIPAL
Authentification, RBAC, journal d'audit, gardes de configuration, caviardage des journaux, ADR 0002 à 0004	Johan
Matrice d'accès et protection des routes, et leurs tests	Johan
Reverse proxy Nginx et TLS, <code>docker-compose.prod.yml</code> , ADR 0007	Johan
Trois environnements, <code>provision-host.sh</code> , <code>deploy.yml</code> , Terraform <code>vm-eni</code> , ADR 0009, 0010 et 0017	Johan
CI unifiée (<code>ci.yml</code>), e2e et k6, supervision, ADR 0014 à 0016	Johan
Certificats DNS-01 et frontal SNI, ADR 0018	Johan
Garage par environnement, rétention, SSE-C, coffre LUKS, ADR 0019 et 0020	Johan, sur une amorce de Valentin
Scan DAST OWASP ZAP (<code>dast.yml</code> , <code>dast-token.sh</code>)	Dorian ; ma part est la revue, trois points bloquants dont une fuite du jeton dans les artefacts
En-tête <code>Cross-Origin-Resource-Policy</code> , module Terraform k3s	Dorian
Workflow frontend, configuration SonarCloud, administration du dépôt	Inès
Ce rapport et les contrôles du 24/09	Johan

10. Usage de l'IA

OUTIL	Claude Code (Anthropic), en assistant dans le terminal
-------	--

TÂCHES	lancement des scans et mise en forme de leurs sorties, recoupement entre la documentation, le code et l'API GitHub, première rédaction de ce rapport
VÉRIFICATIONS HUMAINES	chaque chiffre est lu dans une sortie de preuves / , rejouable par la commande en tête du fichier ; les huit constats gitleaks et les quatre alertes ZAP ont été triés ligne à ligne ; les constats sur la machine viennent de commandes lancées par l'auteur
LIMITE CONSTATÉE	l'outil tend à présenter comme acquis ce qui n'est que prévu, et à conclure avant d'avoir vérifié (une première lecture de l'alerte ZAP citait des codes HTTP non relevés) : d'où les trois marqueurs et le tri ligne à ligne

11. Licences

Outils de ce rapport : Bandit, pip-audit, Checkov, Trivy et OWASP ZAP sous licence Apache 2.0, gitleaks sous licence MIT. Aucun n'est embarqué dans le produit. Composants ajoutés depuis le dossier EC01 : Prometheus, Alertmanager, les exporteurs, cAdvisor, Playwright et boto3 sous Apache 2.0 ; acme.sh sous GPL 3.0 ; **Garage, Grafana et k6 sous AGPL 3.0**. Ils sont utilisés sans modification, chacun dans son propre conteneur, ce qui n'emporte aucune obligation de publication. L'ADR 0019 écarte pourtant MinIO en citant notamment sa licence AGPL, que Garage partage : l'argument ne tient pas, les autres raisons de l'ADR restent. Le recensement complet est dans le rapport collectif EC02.

Annexe • Index des preuves

FICHIER	CONTENU
01-bandit.txt	Bandit, seuil de la CI puis tous niveaux, trois modules
02-pip-audit.txt	pip-audit sur les trois verrous Python
03-npm-audit.txt	npm audit, frontend et tests e2e
04-checkov.txt	Checkov, quatre frameworks
05-gitleaks.txt	gitleaks sur tout l'historique, constats caviardés et tri
06-trivy-config.txt	Trivy config sur le commit
07-trivy-fs.txt	Trivy fs sur l'arbre de travail, clé retirée
08-tests-acces.txt	tests d'accès et d'authentification du backend
09-terraform-validate.txt	terraform fmt et validate sur les deux racines
10-proxy-tls.txt	nginx -t du proxy et du frontal, directives, ports
11-terraform-state.txt	ressources appliquées sur la machine
12-constats-machine.txt	relevés en lecture seule sur la machine
13-github-reglages.txt	environnements, protection des branches, secrets, déploiements
14-dast-zap.txt	dernier rapport OWASP ZAP et tri des alertes

Chaque fichier donne en tête sa date, le commit analysé et la commande exacte pour le rejouer.