LINUX ANALYZER HOSTAUDIT

LOCKHEED LLC

OCTOBRE 2026

Version 0.50

HOSTAUDIT – Linux Analyzer (v0.46.0)

Présentation

hostaudit est un outil d'audit de sécurité Linux **read-only**, conçu pour analyser un serveur de l'intérieur sans le modifier. Il évalue des centaines de contrôles regroupés par domaines (kernel, comptes, SSH, réseau, web, bases de données, secrets, conteneurs, logs, indicateurs de compromission, etc.).

Fonctionnement

Chaque contrôle produit un résultat clair (PASS, FAIL, SKIP ou NA) accompagné de preuves et de recommandations. L'outil calcule un score global ainsi que des notes par axe (Protection, Sécurité, Résilience). Les audits sont stockés en base de données et peuvent être exportés en PDF, Excel, HTML ou JSON.

Évolutions majeures

La version 0.46.0 ajoute le support natif de la famille RPM (RHEL, Rocky, AlmaLinux, Oracle Linux, Fedora, Amazon Linux, SUSE). La base RPM est lue directement, tandis que dnf, yum et zypper ne sont utilisés qu'en mode lecture. Un mode de surveillance continue (Watch), une détection d'intrusion (Sentinel) et des correctifs automatiques sécurisés complètent l'outil.

En résumé

hostaudit combine audit, monitoring, détection et reporting dans un seul produit professionnel, désormais utilisable aussi bien sur les distributions Debian/Ubuntu que sur les systèmes basés sur RPM.

hostaudit

SÉCURITÉ • VISIBILITÉ • CONFORMITÉ

hostaudit – Analyseur de sécurité Linux de niveau professionnel

Une plateforme complète d'audit, de monitoring et de réponse pour sécuriser vos environnements Linux.

Conçu pour les admins.
Pensé pour la sécurité.
Fait pour Linux.

AUDIT

Contrôles de sécurité complets sur des centaines de points (CIS, STIG, best practices).

MONITORING

Surveillance continue en temps réel. Détection des dérives et anomalies.

RÉPONSE

Détection proactive des menaces. Intégration Sentinel pour réaction rapide.



HOSTAUDIT

MOTEUR D'AUDIT & DE SÉCURITÉ LINUX

CORRECTIFS

Auto-fix sûr et sélectif. Scripts idempotents, rollback-ready.

REPORTING

Rapports détaillés, score de sécurité, exports PDF/CSV/JSON.

HISTORIQUE

Traçabilité complète. Comparaison, tendances et évolution.

ARCHITECTURE MONOLITHIQUE DISCIPLINÉE

Modèle de contrôles centralisé, Context sécurisé, politiques strictes.

PUISSANT ET SÛR EN ROOT

Conçu pour être exécuté en root de façon fiable, prévisible et sécurisée.

READ-ONLY BY DESIGN

Aucune modification par défaut. Audit sans altérer vos systèmes.

SCORE & HISTORIQUE

Score de sécurité calculé à chaque exécution. Historique conservé.

EXPORTS & INTÉGRATION

Exports automatisés. API, SIEM, ticketing, orchestration.

MULTI-DISTRIBUTIONS

Fonctionne sur Debian, Ubuntu, RHEL, CentOS, Rocky, Alma et dérivés.

WATCH MODE

Surveillance continue en arrière-plan

SENTINEL

Détection & alerte en temps réel

AUTO-FIX

Correction automatique sécurisée

POLITIQUES STRICTES

Application des règles et exemptions

CONTEXT SÉCURISÉ

Isolation et confinement des données

PROFESSIONNEL & FIABLE

Pensé pour les environnements critiques

hostaudit – Les 14 points clés

1. Outil read-only d'audit Linux depuis l'intérieur, sans modification.

2. Centaines de contrôles regroupés en analyseurs (kernel, comptes, SSH, réseau, web, DB, secrets, conteneurs, logs, IoC...).

3. Résultats clairs : PASS / FAIL / SKIP / NA + preuves et remédiations.

4. Score global + notes par axe (Protection, Sécurité, Résilience) avec notes A–E.

Protection	Sécurité	Résilience
A	B	C

5. Stockage de l'historique (SQLite par défaut, PostgreSQL/ MariaDB/ Oracle).

SQLite · PostgreSQL · MariaDB · Oracle

6. Exports PDF, Excel, HTML, JSON.

PDF · XLSX · HTML · JSON

7. Mode Watch : surveillance continue des drifts + alertes/ corrections.

8. Sentinel : détection temps réel de comportements suspects.

9. Actions de containment réversibles (blocage IP, verrouillage compte, quarantaine...).

Blocage IP · Verrouillage compte · Quarantaine

10. --fix : correctifs automatiques avec backup, test et rollback.

Backup → Test → Rollback

11. Un seul fichier Python 3.12+ + config jamais exécutée.

12. Commandes externes strictement allow-listées (lecture seule).

13. Auto-installation en root dans /usr/local/sbin + planification cron.

14. Conçu pour Ubuntu 24.04/26.04, fonctionne sur d'autres distros avec avertissement.

Ubuntu 24.04 · Ubuntu 26.04

⚠ Autres distros : mode compatible avec avertissement.

Audit. Détection. Correction. Traçabilité. Sans compromis.

AGENDA HOSTAUDIT

- Chapitre 1 – Présentation générale du produit
- Chapitre 2 – Architecture et principes de conception
- Chapitre 3 – Modèle de données et contrôles
- Chapitre 4 – Accès sécurisé à l’hôte (Context)
- Chapitre 5 – Les analyseurs (chapitres fonctionnels)
- Chapitre 6 – Configuration (hostaudit_cfg.py)
- Chapitre 7 – Exécution d’un audit et scoring
- Chapitre 8 – Stockage et historique
- Chapitre 9 – Rapports et exports
- Chapitre 10 – Mode Watch (monitoring continu)
- Chapitre 11 – Sentinel (détection d’intrusion en temps réel)

AGENDA HOSTAUDIT – PART 2

- Chapitre 12 – Réponse et containment (respond)
- Chapitre 13 – Correctifs automatiques (--fix)
- Chapitre 14 – Installation, planification et administration
- Chapitre 15 – Sécurité de l’outil lui-même et limites

Arguments en entrée

Arguments globaux (disponibles sur la plupart des commandes)

Argument	Description
`-u`, `--user`	Utilisateur hostaudit (demandé si omis)
`--password-file PATH`	Lit le mot de passe depuis un fichier privé
`--db URL`	Base de données (`sqlite:///PATH`, `postgresql://`, `mariadb://`, `oracle://`)
`--settings PATH`	Fichier de configuration (défaut : `hostaudit_cfg.py` à côté du programme, puis `/etc/hostaudit/hostaudit_cfg.py`)
`--frames {ascii,unicode}`	Style des tableaux
`--no-color`	Désactive les couleurs
`--local`	Mode test non sécurisé (pas de login, données dans le dossier du programme)

Exports (disponibles sur l'audit et `show`)

Argument	Description
`--pdf [PATH]`	Génère un rapport PDF
`--excel [PATH]`	Génère un classeur Excel
`--html [PATH]`	Génère un rapport HTML autonome
`--json [PATH]`	Génère un rapport JSON

Argument	Description
`--analyzer CODES`	Exécute uniquement ces analyseurs (`LOG` , `APP,WEB` , `--analyzer list` pour la liste)
`--only IDS`	Exécute uniquement ces contrôles (préfixes d'ID, ex. `NET` ou `NET-005`)
`--skip IDS`	Exclut ces contrôles
`--fs-budget SECONDS`	Limite de temps du scan du système de fichiers (défaut : 120)
`--secret`	Scan profond des secrets dans le code
`--discover`	Inventaire du code par langage (sans évaluation des contrôles)
`--lang LANGUAGES`	Langages pour l'analyseur COD (`python` , `php,sh` , etc.)
`--expected-tcp PORTS`	Ports TCP autorisés à être exposés (défaut : 22,80,443)
`--fix`	Applique les correctifs automatiques après l'audit
`--dry-run`	Avec `--fix` : montre les changements sans les appliquer
`--yes`	Avec `--fix` : applique sans demander confirmation

`--no-restart`	Avec `--fix` : exclut les correctifs qui redémarrent un service
`--fail-on {never,low,medium,high,critical}`	Code de sortie 1 si un échec atteint cette sévérité (défaut : high)
`-v`, `--verbose`	Détaille aussi les contrôles PASS et NA
`--critical` / `--high` / `--medium`	Filtre le rapport par sévérité minimale
`-q`, `--quiet`	Une seule ligne de résultat (pour les audits planifiés)
`--summary`	Tableau de bord synthétique
`-s`, `--short`	Rapport condensé (score + échecs critiques/high)
`--no-db`	Audit ponctuel sans login ni stockage
`--no-progress`	Désactive la barre de progression
`--list`	Liste tous les contrôles et quitte
`--self-test`	Exécute les tests internes et quitte
`--version`	Affiche la version

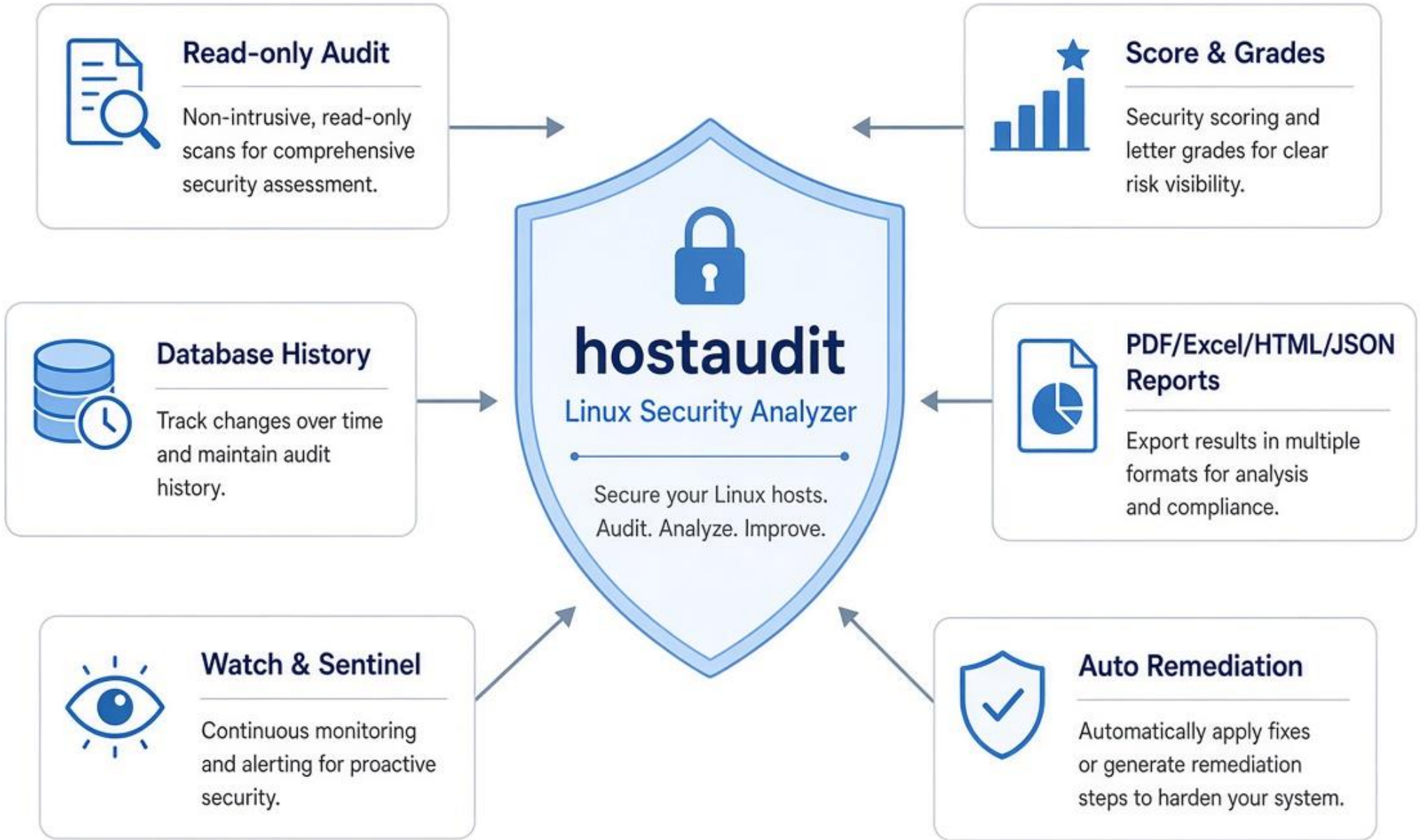
Sous-commandes

Commande	Arguments principaux
user	`add NAME [--admin]`, `list`, `passwd [NAME]`, `delete NAME [--with-audits]`
history	`--all-users`, `--host`, `--limit`
show	`AUDIT` (défaut : `last`), plus les options d'export et de filtre
secrets	`list` / `show` / `bench` + options
inventory	`list` / `show`
feeds	`inventory` / `fetch` / `show`
fixes	`list` / `show` / `undo`
paths	Analyse des chemins d'attaque (`--to`, `--from`, `--all`, `--json`)
respond	`list`, `show`, `policy`, `scan`, `accept`, `arm`, `disarm`, `undo`, `keep`, `freeze`, `block`, `lock`, `quarantine`, `pause`
watch	`status`, `run`, `once`, `events`, `policy`, `accept`, `pause`, `resume`, `verify`, `install`, `remove`
schedule	`show`, `install`, `remove`
settings	`init [PATH] [--force]`, `show`
install	(aucun) ou `status`
sbom	Génération de SBOM
help	`[TOPIC]` (guide intégré)

Notes importantes

- Toute option en ligne de commande **écrase** le paramètre correspondant du fichier `hostaudit_cfg.py`.
- Les options `--local` et `--no-db` désactivent l'authentification et le stockage.
- Le mode `--fix` est toujours protégé contre la perte d'accès distant (confirmation, test, rollback).
- La commande `watch` et `respond` peuvent fonctionner en mode service (systemd).

1 - Présentation Générale


Read-only Audit
Non-intrusive, read-only scans for comprehensive security assessment.
Score & Grades
Security scoring and letter grades for clear risk visibility.
hostaudit
Linux Security Analyzer
Secure your Linux hosts.
Audit. Analyze. Improve.
Database History
Track changes over time and maintain audit history.
PDF/Excel/HTML/JSON Reports
Export results in multiple formats for analysis and compliance.
Watch & Sentinel
Continuous monitoring and alerting for proactive security.
Auto Remediation
Automatically apply fixes or generate remediation steps to harden your system.

Chapitre 1 – Présentation générale du produit

hostaudit – AUDIT INTERNE EN LECTURE SEULE –

CENTAINES DE CONTRÔLES SÉCURITÉ

- **Kernel Hardening** — sysctls, modules, spectre/meltdown
- **Comptes & Privilèges** — utilisateurs, groupes, sudo, PAM
- **SSH & Accès Distants** — sshd_config, clés, authentification
- **Réseau & Firewall** — netfilter, ports, services, interfaces
- **Web (Apache/Nginx)** — configurations, permissions, headers
- **Bases de Données** — PostgreSQL, MySQL, MariaDB, accès
- **Secrets & Sensibles** — clés, mots de passe, tokens, fichiers
- **Conteneurs & Runtime** — Docker, Podman, CRI-O, runc
- **Logs & Journal** — syslog, journald, auditd, rotations
- **IoC & Indicateurs** — fichiers suspects, processus, hashes

CALCUL DU SCORE
Score de sécurité pondéré (0–100) + niveaux de criticité

HISTORIQUE STOCKÉ
Résultats et scores conservés dans une base locale en lecture seule.

EXPORTS DE RAPPORTS
Générez des rapports dans les formats de votre choix.

TOOL EN LECTURE SEULE – AUCUNE MODIFICATION APPORTÉE À VOTRE SYSTÈME

DISTRIBUTIONS CIBLES

- **Ubuntu 24.04 LTS** — Noble Numbat
- **Ubuntu 26.04 LTS** — Plucky Puffin

AUTRES DISTRIBUTIONS
Possible, mais non qualifié en standard

PRISE EN CHARGE NON GARANTIE

TECHNOLOGIE

Python 3.12+
bibliothèque standard uniquement

FICHIER UNIQUE
hostaudit
+ fichier de configuration
hostaudit_cfg.py

LU ET ANALYSÉ UNIQUEMENT
(JAMAIS EXÉCUTÉ)

PHILOSOPHIE

LECTURE PURE DU SYSTÈME
Aucune altération, aucun changement, aucun impact sur le système audité.

COMMANDES EXTERNES AUTORISÉES
Liste blanche stricte, mode listing uniquement, utilisées uniquement pour collecter des informations.

CONTRÔLE NON ÉVALUABLE = SKIP
Jamais considéré comme PASS.
Transparence et honnêteté avant tout.

AUDITER. COMPRENDRE. SÉCURISER. SANS JAMAIS MODIFIER.

Chapitre 1 – Présentation générale du produit

hostaudit est un outil d'audit de sécurité **read-only** d'un hôte Linux, vu de l'intérieur.

Il évalue des centaines de contrôles (hardening kernel, comptes, SSH, réseau, web, bases de données, secrets, containers, logs, IoC, etc.), calcule un score, stocke l'historique et peut exporter des rapports (PDF, Excel, HTML, JSON).

Cibles principales : **Ubuntu 24.04 / 26.04** (autres distributions avec avertissement).

Il tourne en Python 3.12+ (standard library uniquement, sauf exports PDF/Excel optionnels).

Un seul fichier source + un fichier de configuration `hostaudit_cfg.py` (jamais exécuté, seulement parsé).

Philosophie :

- Lecture pure du système (sauf base de données et rapports de l'outil lui-même).
- Commandes externes strictement allow-listées et en mode listing uniquement.
- Contrôle qui ne peut pas être évalué → SKIP (jamais PASS).

(hostaudit v0.50.0)

hostaudit est un outil d'audit de sécurité **read-only** d'un hôte Linux, vu de l'intérieur. Il évalue des centaines de contrôles (hardening kernel, comptes, SSH, réseau, web, bases de données, secrets, conteneurs, Kubernetes, logs, indicateurs de compromission, etc.), calcule un score, stocke l'historique et peut exporter des rapports (PDF, Excel, HTML, JSON).

Cibles principales

Ubuntu 24.04 / 26.04 (testées). Depuis la version 0.46, la famille RPM est supportée nativement (RHEL, Rocky, AlmaLinux, Oracle Linux, Fedora, Amazon Linux, SUSE). Les autres distributions tournent avec un avertissement.

Environnement technique

L'outil tourne en Python 3.12+ (standard library uniquement). Les exports PDF et Excel nécessitent reportlab et openpyxl, installés automatiquement dans un répertoire privé de l'outil lorsqu'ils sont absents. Il se présente sous la forme d'un seul fichier source (`hostaudit.py`) accompagné d'un fichier de configuration (`hostaudit_cfg.py`) qui est **parsé et jamais exécuté**.

Philosophie de conception

- Lecture pure du système : l'outil n'écrit que sa propre base de données, les rapports demandés et, éventuellement, un fichier cron lors de l'installation de la planification.
- Commandes externes strictement allow-listées et limitées à des opérations de listing uniquement.
- Un contrôle qui ne peut pas être évalué (absence de root, outil manquant, etc.) est systématiquement reporté en **SKIP**, jamais en PASS.
- Séparation claire entre le code et la configuration : toute option en ligne de commande prime sur le fichier de settings.

Évolutions récentes (v0.50)

La version 0.50 introduit un analyseur **Kubernetes** (K8S), des mécanismes de scellement des rapports (clé HMAC + option Ed25519), ainsi que de nouvelles commandes orientées flotte et conformité (`fleet` , `compliance` , `alerts` , `serve` , `push` , `verify`). Le modèle de données reste compatible avec des audits provenant d'un analyseur Windows (chapitres RDP, UPD, DEF).

En résumé, hostaudit est conçu comme un produit professionnel autonome : il combine audit, scoring, historique, surveillance continue, détection et remédiation, tout en restant volontairement minimaliste dans ses dépendances et strict dans ses accès au système.

2 – Architecture et principes de conception

ARCHITECTURE MONOLITHIQUE HAUTEMENT STRUCTURÉE

MODÈLE CENTRAL
Check → Outcome → Finding

Un modèle unifié pour tous les contrôles, résultats et conclusions.

REGISTRY GLOBAL DES CONTRÔLES
Référentiel unique de tous les analyseurs et contrôles

Déclaration, métadonnées, dépendances et configuration centralisées.

CONTEXTE SÉCURISÉ (Couche d'accès)
Accès en lecture seule aux hosts avec cache intelligent

Fichiers · sysctl · Commandes · Cache court & sécurisé · Isolation totale.

MODÈLE CENTRAL



CHECK
Contrôle exécuté avec contexte sécurisé

OUTCOME
Résultat structuré et normalisé

FINDING
Conclusion actionable avec sévérité et justification

RÉSULTATS UNIFIÉS



Sévérité (Critique / High / Medium / Low / Info)

Justification & preuves

Références & remédiation

Traçabilité & métadonnées

ANALYSEURS (CHAPTERS)
Modules spécialisés · Couverture exhaustive · Registry central

KRN	Encryption	Mandatory Access Control	Packages	Accounts
SSH	MFA	NET	FS	PRM
SVC	WEB	TLS	DB	SEC
APP	PHP	COD	CTR	LOG
IOC	PRC	TRC	BKP	CLD

AXES DE QUALITÉ
Évaluation structurée en 3 dimensions

PROTECTION
Réduction de la surface d'attaque et durcissement.
Niveau cible : A / B — A

SÉCURITÉ
Prévention des failles et des mauvaises configurations.
Niveau cible : A / B — B

RÉSILIENCE
Détection, maintien et capacité à réagir.
Niveau cible : A / B — B

SYSTÈME DE NOTATION A–E
Plafonnement selon les défaillances critiques/high

- A Excellent
- B Bon
- C Acceptable
- D Faible
- E Critique

Les défaillances critiques ou high plafonnent automatiquement la note selon des seuils prédéfinis.

RÈGLES DE CONCEPTION STRICTES

AUCUNE ÉCRITURE SUR LE SYSTÈME AUDITÉ
Sauf sa propre base de données et ses tâches cron dédiées.

COMMANDES EXTERNES CONTRÔLÉES
Liste d'autorisation stricte + politique d'arguments.

LECTURE DE FICHIERS ENCADRÉE
Taille limitée, types réguliers uniquement, non bloquant.

INSTALLATION MAÎTRISÉE
Auto-installation dans /usr/local/sbin + /etc/hostaudit lorsque exécuté en root.

Chapitre 2 – Architecture et principes de conception

Architecture monolithique mais très structurée :

- **Modèle central** : `Check` → `Outcome` → `Finding`.
- **Registry** global de tous les contrôles.
- **Context** : couche d'accès sécurisée et mise en cache à l'hôte (fichiers, sysctl, commandes, etc.).
- **Chapitres / Analyseurs** (KRN, ENC, MAC, PKG, ACC, SSH, MFA, NET, FS, PRM, SVC, WEB, TLS, DB, SEC, APP, PHP, COD, CTR, LOG, IOC, PRC, TRC, BKP, CLD).
- Trois axes de qualité : **Protection**, **Sécurité**, **Résilience**.
- Système de notes (A–E) avec plafonds en cas d'échec critique/high.

Règles de design strictes :

- Jamais d'écriture sur le système audité (sauf base de données propre et cron).
- Commandes externes limitées à une liste blanche + politique d'arguments.
- Lecture de fichiers bornée (taille, type régulier uniquement, non-blocking).
- Installation auto en `/usr/local/sbin` + `/etc/hostaudit` quand lancé en root.

Architecture générale

hostaudit repose sur une architecture monolithique volontairement simple : un seul fichier Python de plusieurs milliers de lignes. Cette approche évite les dépendances complexes et facilite le déploiement (un seul binaire à copier). Malgré cela, le code est fortement structuré autour de quelques concepts centraux qui garantissent la lisibilité et la maintenabilité.

Modèle de données central

Le cœur du système est le triplet **Check → Outcome → Finding** :

- Un **Check** est la définition d'un contrôle (identifiant stable, titre, sévérité, remédiation, référence, besoin de root).
- L'exécution d'un Check produit un **Outcome** (statut PASS / FAIL / SKIP / NA / ERROR + preuves + note).
- L'association des deux forme un **Finding**, qui est l'unité stockée et affichée dans les rapports.

Tous les contrôles sont enregistrés dans un **Registry** global au démarrage du programme. Ce registre permet de lister, filtrer et exécuter les contrôles de façon dynamique.

Couche d'accès à l'hôte : Context

La classe **Context** constitue la seule porte d'entrée vers le système audité. Elle encapsule :

- la lecture sécurisée des fichiers (réguliers uniquement, non bloquante, bornée en taille),
- l'exécution de commandes externes strictement allow-listées,
- la mise en cache des résultats pour éviter les lectures multiples,
- la gestion des inventaires (clés SSH, modules, etc.) pour les comparaisons inter-audits.

Aucune autre partie du code n'accède directement au système de fichiers ou n'exécute de commandes.

Organisation par analyseurs (chapitres)

Les contrôles sont regroupés en **analyseurs** identifiés par des codes courts (KRN, ENC, MAC, PKG, ACC, SSH, MFA, NET, FS, PRM, SVC, WEB, TLS, DB, SEC, APP, PHP, COD, CTR, K8S, LOG, IOC, PRC, TRC, BKP, CLD). Chaque analyseur correspond à un domaine fonctionnel précis. Cette organisation permet de lancer un audit partiel (`--analyzer SSH,NET`) ou de filtrer les rapports.

Système de scoring et d'axes

Trois axes de qualité transverses structurent l'évaluation :

- **Protection** : ce qui empêche un attaquant d'entrer,
- **Sécurité** : ce qui limite les dégâts une fois à l'intérieur,
- **Résilience** : ce qui permet de détecter, de rester à jour et de se remettre.

Règles de design strictes

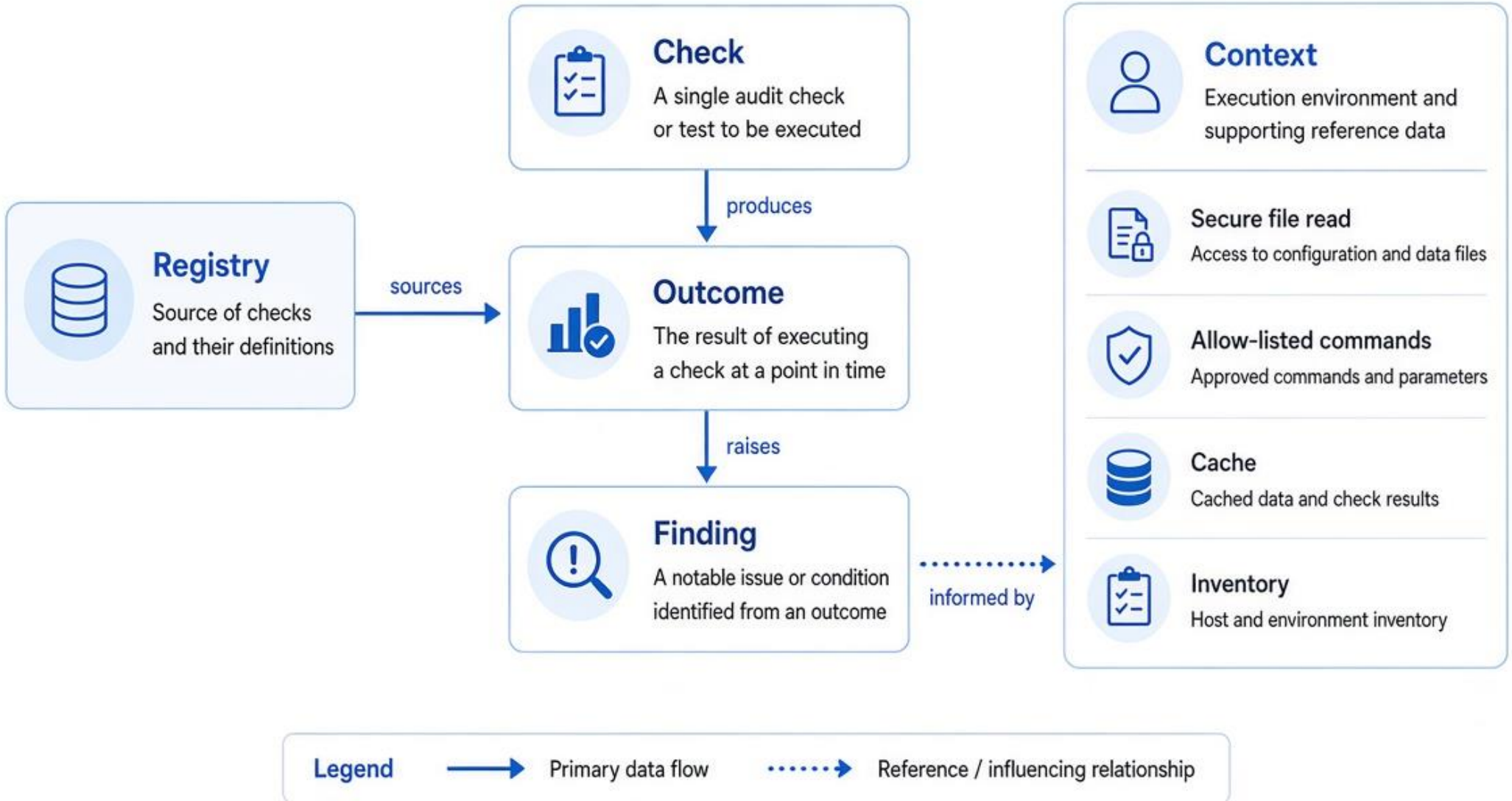
Plusieurs règles de conception sont appliquées de façon systématique :

- **Read-only strict** : aucune écriture n'est effectuée sur le système audité (sauf base de données propre et fichier cron).
- **Allow-list de commandes** : seules les commandes listées dans `ALLOWED_COMMANDS` peuvent être exécutées, et uniquement avec des arguments conformes à une politique de lecture pure.
- **Lecture bornée et sécurisée** : les fichiers sont ouverts en mode non-bloquant, seuls les fichiers réguliers sont acceptés, et la taille lue est limitée.
- **Installation automatique** : lorsqu'il est lancé en root, l'outil s'installe de lui-même dans `/usr/local/sbin` et place sa configuration dans `/etc/hostaudit`, tout en vérifiant l'intégrité de la copie installée.

Ces principes garantissent que l'outil reste prévisible, auditable et sûr à exécuter avec les privilèges root.

HostAudit Core Data Model

Key entities and relationships



3 – Modèle de données et contrôles

hostaudit

AUDIT DE CONFIGURATION HÔTE

Chapitre 3 – Modèle de données et contrôles

Structure des contrôles, classification par axe et calcul des scores

```
// Un contrôle est un objet Check
Check
• id : string  // ex. KRN-001, WEB-007
• title : string
• severity : Severity  // INFO → CRITICAL
• remediation : string
• reference : string  // ex. CIS, ANSSI, STIG...
• needs_root : bool
evaluate(ctx Context) Outcome
// Fonction pure : Context → Outcome
```

reçoit un Context

```
Context
  • hostname
  • os_info
  • packages
  • processes
  • sysctl
  • files
  • ...
```

Outcome (résultat d'un contrôle)

```
status: Status
```

PASS | FAIL | SKIP | NA | ERROR

```
evidence : string // preuves collectées
note : string? // note additionnelle optionnelle
```

retourne un Outcome

severities

- INFO → (observation)
- LOW → (faible)
- MEDIUM → (moyen)
- HIGH → (élevé)
- CRITICAL – (critique)

ORGANISATION DES CONTRÔLES

GROUPÉS PAR CHAPITRE (dictionary CHAPTERS)

```
CHAPTERS = {
  "01-base" :  : { name: "Base système", controls:       [KRN-001, KRN-002, ...] }
  "02-network" : { name: "Réseau et services", controls: [NET-001, NET-002, ...] }
  "03-web" :   : { name: "Services web", controls:       [WEB-001, WEB-002, ...] }
  "04-privileges" { name: "Privilèges et accès", controls: [PRV-001, PRV-002, ...] }
  ...
```

Chaque chapitre regroupe des contrôles logiquement liés.

CLASSÉS PAR AXE DE PROTECTION

PROTECTION
Contrôles préventifs de durcissement et de configuration sécurisée.

SÉCURITÉ
Contrôles de détection, authentification, autorisation et journalisation.

RÉSILIENCE
Contrôles de redondance, sauvegarde, tolérance aux pannes et reprise.

Un contrôle appartient à un chapitre ET contribue à un axe.

CALCUL DES SCORES

RÈGLES DE SCORE

Les scores sont calculés à partir des poids de sévérité et des plafonds (cells).

- INFO : 0
- LOW : 1
- LOW : 1
- MEDIUM : 3
- HIGH : 7
- CRITICAL : 12

– Plafonds (effet non corrigé)
- CRITICAL non corrigé → plafond D
- HIGH non corrigé → plafond B

Les scores sont bornés et les plafonds s'appliquent avant agrégation.

SCORE GLOBAL

Score agrégé sur l'ensemble des contrôles (pondéré par sévérité et plafonné).



Excellent

NOTES PAR AXE

Score calculé par axe selon les mêmes règles (poids + plafonds).



Les scores aident à prioriser les actions ; la revue des résultats reste essentielle.

Chapitre 3 – Modèle de données et contrôles

Chaque contrôle est un objet `Check` :

- `id` (ex. `KRN-001`, `WEB-007`)
- `title`, `severity` (INFO → CRITICAL)
- `remediation`, `reference`
- `needs_root`
- fonction pure qui reçoit un `Context` et retourne un `Outcome`

`Outcome` : `status` (PASS / FAIL / SKIP / NA / ERROR) + evidence + note.

Les contrôles sont regroupés par **chapitre** (dictionnaire `CHAPTERS`) et classés sur un **axe** (protection / security / resilience).

Le score global et les notes par axe sont calculés à partir des poids de sévérité et des plafonds (un CRITICAL non corrigé plafonne à D, un HIGH à B).

Définition d'un contrôle

Chaque contrôle est représenté par un objet **Check** immuable. Cet objet contient toutes les métadonnées nécessaires à l'exécution et à la restitution du résultat :

- un **identifiant stable** (ex. `KRN-001`, `WEB-007`) qui ne change jamais d'une version à l'autre,
- un **titre** descriptif,
- une **sévérité** (INFO, LOW, MEDIUM, HIGH, CRITICAL),
- un texte de **remédiation** et une éventuelle **référence** (CIS, NIST, etc.),
- un indicateur `needs_root` qui précise si le contrôle nécessite les privilèges root,
- une **fonction pure** qui reçoit un objet `Context` et retourne un `Outcome`.

Cette fonction est volontairement pure : elle ne modifie rien et ne dépend d'aucun état global hors du `Context` qui lui est passé.

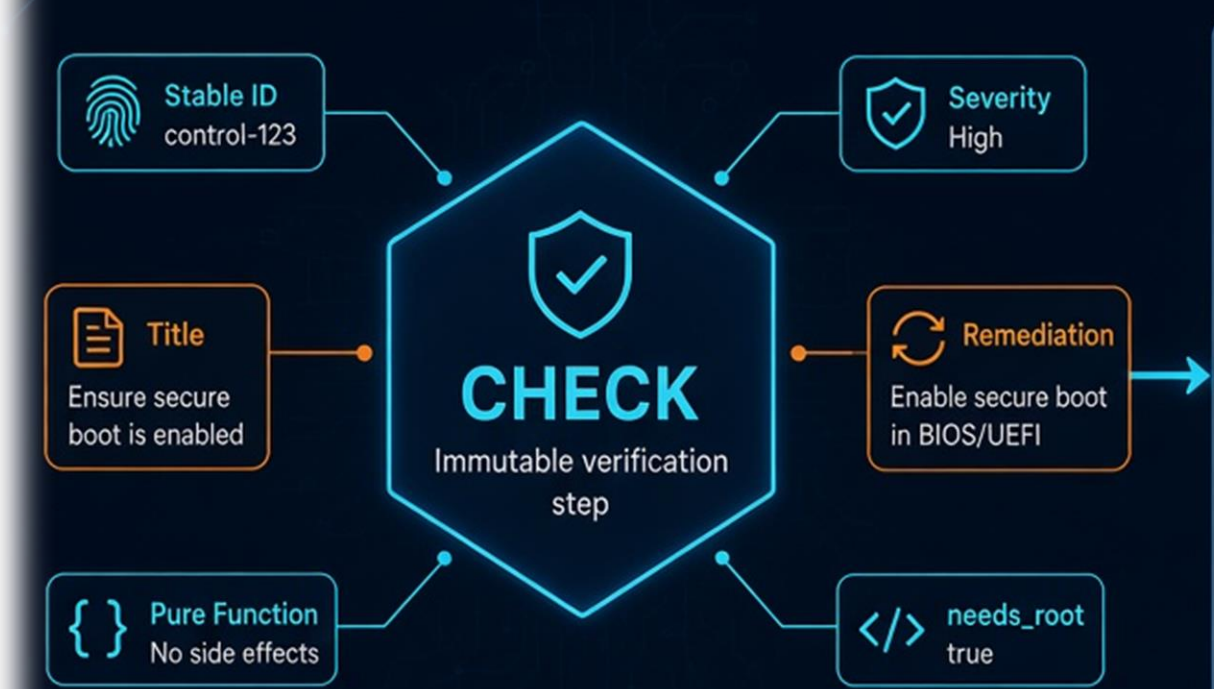
Résultat d'un contrôle : Outcome

L'exécution d'un Check produit un **Outcome** qui contient :

- un **statut** parmi cinq valeurs possibles :
  PASS (conforme), **FAIL** (non conforme), **SKIP** (impossible à évaluer), **NA** (non applicable), **ERROR** (le contrôle lui-même a planté),
- une liste de **preuves** (evidence) qui justifient le résultat,
- une **note** optionnelle.

Definition of a Control

Immutable Check → Outcome



OUTCOME

Result of applying the check.

STATUS

PASS | FAIL | SKIP | NA | ERROR

EVIDENCE

- Command / probe output
- File / registry / system state
- Logs / events
- Metadata & context

OPTIONAL NOTE

Human-readable explanation or context when needed

Controls are immutable, deterministic, and produce reproducible outcomes.

Data / Reference — Action / Remediation — System Context

Organisation par chapitres et axes

Les contrôles sont regroupés par **chapitre** via un dictionnaire `CHAPTERS` (KRN, SSH, NET, WEB, etc.). Chaque chapitre correspond à un domaine fonctionnel.

Parallèlement, chaque contrôle est rattaché à un **axe de qualité** :

- **Protection** : empêche l'attaquant d'entrer,
- **Sécurité** : limite les dégâts une fois à l'intérieur,
- **Résilience** : permet de détecter, de rester à jour et de se remettre.

Le rattachement se fait soit automatiquement via l'analyseur, soit explicitement via une table de surcharge (`CONTROL_AXIS`).

Calcul du score et des notes

Le score global est calculé à partir des poids associés à chaque sévérité (CRITICAL = 10, HIGH = 7, MEDIUM = 4, LOW = 1). Les contrôles en SKIP ou NA ne sont pas pris en compte.

Une note alphabétique (A à E) est ensuite attribuée selon des seuils fixes (A ≥ 90, B ≥ 75, etc.). Des **plafonds** sont appliqués :

- la présence d'au moins un échec **CRITICAL** plafonne la note à **D**,
- la présence d'au moins un échec **HIGH** plafonne la note à **B**.

Organization by Chapters and Axes

Controls organized by chapters and quality axes. Scored by severity with letter-grade outcomes.

CHAPTERS

Groupings of controls by domain

KRN
Kernel & System
28 controls

SSH
Secure Shell
20 controls

NET
Network & Firewall
24 controls

WEB
Web & Application
18 controls

Chapters group related controls by functional domain.

QUALITY AXES

Evaluate control effectiveness across 3 dimensions

1. PROTECTION
Prevent entry

Stop threats and unauthorized access before they reach the system.

2. SECURITY
Limit damage

Contain impact, reduce attack surface, and prevent lateral movement.

3. RESILIENCE
Detect, stay current, recover

Detect issues, keep systems updated, and recover quickly.

Axes measure how well controls protect, contain, and sustain security.

SCORE CALCULATION

Severity-weighted scoring with caps and letter grade

1. SEVERITY WEIGHTS

CRITICAL	10
HIGH	7
MEDIUM	4
LOW	1

2. RAW SCORE

Sum of (Finding Severity Weight) across all failed controls.

3. LETTER GRADE (EXAMPLE)

A	B	C	D	E
≥ 90	75 – 89	50 – 74	25 – 49	< 25

4. GRADE CAPS (FAILURE RULES)

- CRITICAL failure caps grade at D
- HIGH failure caps grade at B

Caps reflect real risk: unresolved critical or high-severity issues limit the overall grade.

SCORE FORMULA Raw Score = ∑ (Severity Weight of each failed control) → Apply Grade Caps (if any) → Map to Letter Grade (A–E)

Lower score = stronger security

Calcul du score et des notes

Le score global est calculé à partir des poids associés à chaque sévérité (CRITICAL = 10, HIGH = 7, MEDIUM = 4, LOW = 1). Les contrôles en SKIP ou NA ne sont pas pris en compte.

Une note alphabétique (A à E) est ensuite attribuée selon des seuils fixes (A ≥ 90, B ≥ 75, etc.). Des **plafonds** sont appliqués :

- la présence d'au moins un échec **CRITICAL** plafonne la note à **D**,
- la présence d'au moins un échec **HIGH** plafonne la note à **B**.

Ce mécanisme garantit qu'un seul problème grave suffit à empêcher une bonne note globale, même si le reste du système est conforme.

Score and Grade Calculation

CONTROL EFFECTIVENESS SCORING AND GRADING LOGIC

1. SEVERITY WEIGHTS

Each failed control is assigned a severity weight.

CRITICAL	Critical impact, widespread risk	10
HIGH	Significant impact, broad risk	7
MEDIUM	Moderate impact, limited risk	4
LOW	Low impact, minimal risk	1

Total Score = Sum of weights for all failed controls

2. GRADE THRESHOLDS

Final score is mapped to a letter grade.

A	≥ 90	Excellent
B	≥ 75	Good
C	≥ 55	Fair
D	≥ 30	Poor
E	< 30	Very Poor

3. GRADE CAPS: ONE SERIOUS ISSUE IS ENOUGH

A single serious failure limits the maximum possible grade.

- ANY CRITICAL FAILURE caps grade at **D**
- ANY HIGH FAILURE caps grade at **B**
- ONE SERIOUS ISSUE CAN PREVENT A GOOD OVERALL GRADE

STRONGER CONTROLS. BETTER SCORE. HIGHER GRADE.

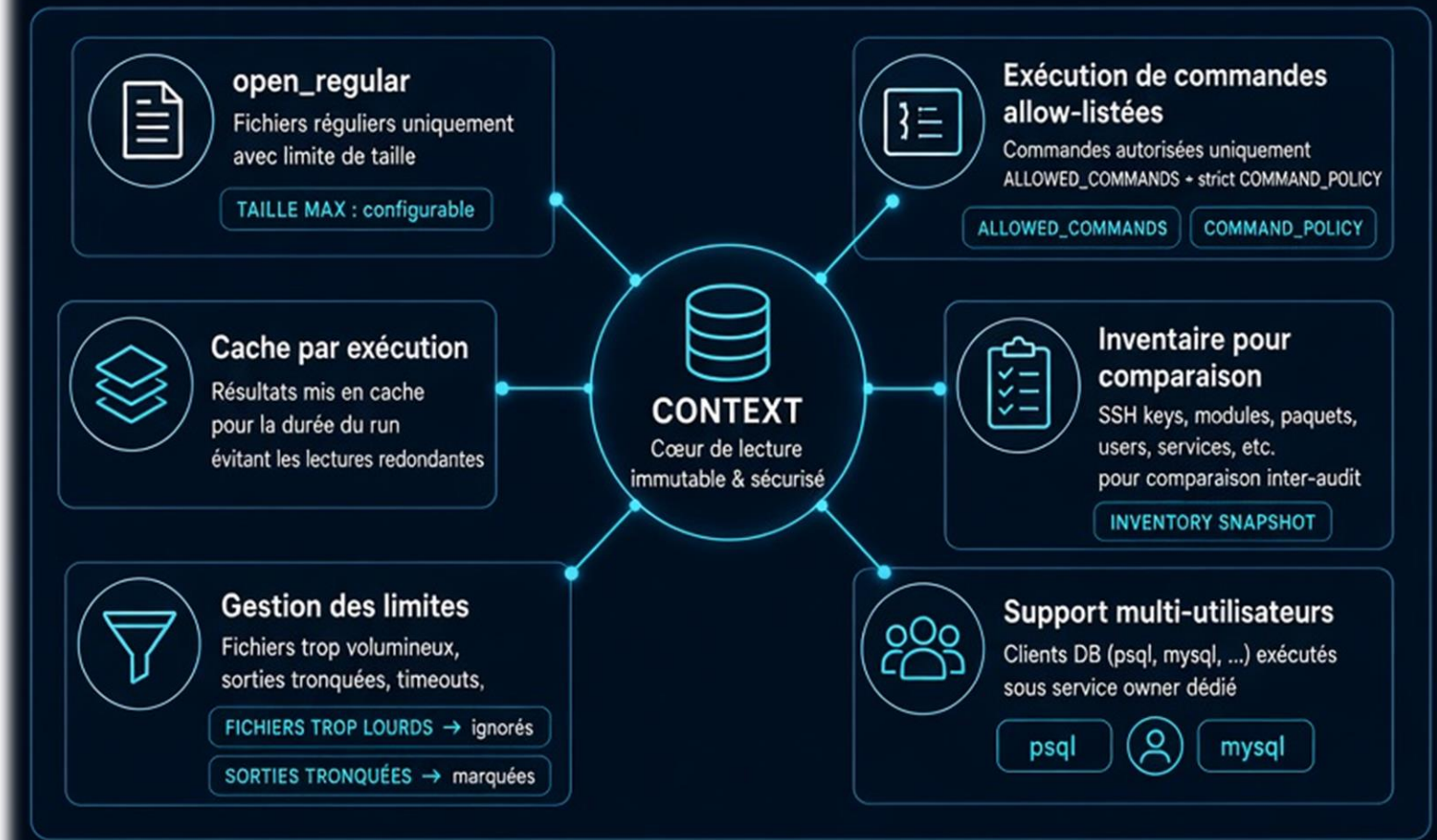
ASSESS. SCORE. IMPROVE.

4 – Accès sécurisé à l’hôte (Context)

hostaudit
AUDIT DE CONFORMITÉ DES HÔTES

CYBER SÉCURITÉ & CONFORMITÉ

Chapitre 4 – Accès sécurisé à l'hôte (Context)



POLITIQUE DE SÉCURITÉ FORTE

Aucune commande ne peut modifier le système
Exécution en lecture seule, sans effet de bord.

Aucun autoload de module kernel via firewall
Blocage explicite via policy et confinement

Aucune requête SQL avec effets de bord
Requêtes en lecture seule, sans INSERT/UPDATE/DELETE

Principe du moindre privilège
Comptes dédiés, droits minimaux et isolation stricte.

Conçu pour auditer, jamais pour altérer.

Le Context est la racine de toute lecture : tout commence ici, tout reste traçable, rien ne modifie.

SÉCURISÉ • PRÉDICTIBLE • AUDITABLE

Chapitre 4 – Accès sécurisé à l'hôte (Context)

La classe `Context` est le cœur de la lecture :

- Lecture de fichiers réguliers uniquement (`open_regular` + limite de taille).
- Exécution de commandes allow-listées (`ALLOWED_COMMANDS` + `COMMAND_POLICY` très strictes).
- Cache par run.
- Inventaire (clés SSH, modules, etc.) pour comparaison inter-audits.
- Gestion des limites (fichiers trop gros, sorties tronquées).
- Support multi-utilisateurs pour clients DB (psql/mysql en tant que propriétaire du service).

Politique de sécurité forte :

aucune commande ne peut modifier le système, pas d'autoload de modules kernel via firewall, pas de requêtes SQL avec effets de bord, etc.

Rôle de la classe Context

La classe **Context** est la seule couche autorisée à interagir avec le système audité. Elle centralise toutes les opérations de lecture et garantit que les règles de sécurité sont appliquées de façon uniforme, quel que soit le contrôle qui les invoque. Aucun autre composant du code n'ouvre de fichier ni n'exécute de commande directement.

Lecture sécurisée des fichiers

La lecture de fichiers passe exclusivement par des fonctions dédiées (`open_regular` , `read_head` , `read`). Plusieurs protections sont appliquées :

- seuls les **fichiers réguliers** sont acceptés (pas de FIFO, pas de devices, pas de sockets),
- l'ouverture est **non bloquante** afin d'éviter tout blocage sur un fichier spécialement préparé,
- la taille lue est **bornée** (paramètre `READ_MAX_MB`),
- les fichiers trop volumineux sont tronqués et signalés dans le rapport.

Les logs sont lus de façon streamée avec des limites supplémentaires (taille totale et longueur de ligne) pour éviter les bombes de compression ou les fichiers sans saut de ligne.

Exécution de commandes allow-listées

Seules les commandes listées dans `ALLOWED_COMMANDS` peuvent être exécutées. Pour chacune d'elles, une **politique d'arguments** (`COMMAND_POLICY`) vérifie que l'appel reste strictement en mode lecture :

Cache, inventaires et limites

Le `Context` maintient un **cache par run** afin d'éviter de relire plusieurs fois les mêmes sources. Il gère également des **inventaires** (clés SSH, modules chargés, etc.) qui sont comparés d'un audit à l'autre pour détecter les changements.

Toutes les limites atteintes (fichiers tronqués, sorties coupées, timeouts) sont enregistrées et mentionnées dans le rapport final, afin que l'auditeur sache précisément ce qui n'a pas pu être entièrement analysé.

Support multi-utilisateurs pour les bases de données

Lorsqu'un contrôle interroge une base de données (PostgreSQL, MySQL/MariaDB…), le `Context` peut exécuter le client (`psql` , `mysql` , etc.) sous l'identité du propriétaire du service. Cela permet de respecter les droits Unix sans donner de privilèges excessifs au processus d'audit.

Politique de sécurité globale

L'ensemble de ces mécanismes garantit qu'aucune commande ne peut modifier le système, qu'aucun module kernel n'est chargé de façon détournée, et qu'aucune requête SQL ne peut avoir d'effet de bord. Le `Context` constitue ainsi une frontière de confiance claire entre le code d'audit et l'hôte analysé.

5 – Les analyseurs (chapitres fonctionnels)

hostaudit

AUDIT DE SÉCURITÉ LINUX

OUTIL D'AUDIT DE SÉCURITÉ POUR LES SYSTÈMES LINUX | ANALYSE • ÉVALUE • RENFORCE

VISIBILITÉ • CONFORMITÉ • SÉCURITÉ

KRN durcissement noyau

- ASLR
- ptrace
- BPF
- mitigations CPU
- Secure Boot
- lockdown
- core dumps
- modules inutiles

ENC Chiffrement au repos

- LUKS
- GRUB
- modes de secours

MAC Contrôle d'accès obligatoire

- AppArmor
- SELinux

PKG Mises à jour & vulnérabilités

- Mises à jour
- CVEs (Ubuntu Pro + OSV)
- Dépendances applicatives

ACC Comptes & accès

- Comptes
- UID 0
- sudo
- PAM
- comptes dormants

SSH Configuration SSH

- Configuration sshd
- Clés
- Historiques de connexion

MFA Second facteur

- sshd
- sudo
- su

NET Réseau & exposition

- Ports exposés
- firewall
- routes
- Docker publish
- sysctl réseau

FS / PRM Permissions & durcissement

- Permissions
- setuid
- capabilities
- fichiers sensibles

SVC Services & planification

- Services systemd
- timers
- linger

WEB Serveurs web

- nginx / Apache
- TLS
- headers
- vhosts
- path traversal
- méthodes

TLS Chiffrement & certificats

- Certificats
- OpenSSL policy
- DH
- mail
- HAProxy

DB Bases de données

- PostgreSQL
- MySQL / MariaDB
- Redis
- MongoDB
- (live + fichiers)

SEC Secrets & fuites

- Secrets lisibles
- clés, tokens
- historique shell

AUDIT COMPLET • SÉCURITÉ RENFORCÉE

Identifiez les risques, appliquez les bonnes pratiques et démontrez la conformité de vos systèmes Linux.

LÉGER & NATIF | RAPPORTS DÉTAILLÉS | CONÇU POUR LES ADMINS

OPEN SOURCE & MODULAIRE | FONCTIONNE SANS AGENT | SORTIE JSON / HTML | https://hostaudit.io

Code	Périmètre
KRN	Hardening kernel (ASLR, ptrace, BPF, mitigations CPU, Secure Boot, lockdown, core dumps, modules inutiles…)
ENC	Chiffrement au repos, LUKS, GRUB, modes de secours
MAC	AppArmor / SELinux
PKG	Mises à jour, CVEs (Ubuntu Pro + OSV), dépendances applicatives
ACC	Comptes, UID 0, sudo, PAM, comptes dormants
SSH	Configuration sshd, clés, historiques de connexion
MFA	Second facteur (sshd/sudo/su)
NET	Ports exposés, firewall, routes, Docker publish, sysctl réseau
FS / PRM	Permissions, setuid, capabilities, fichiers sensibles
SVC	Services systemd, timers, linger
WEB	nginx/Apache (TLS, headers, vhosts, path traversal, méthodes…)
TLS	Certificats, OpenSSL policy, DH, mail, HAProxy
DB	PostgreSQL, MySQL/MariaDB, Redis, MongoDB… (live + fichiers)
SEC	Secrets lisibles (clés, tokens, historique shell…)

APP / PHP / COD	Frameworks, PHP2Shell, code exposé (injection, debug, désérialisation…)
CTR	Docker/Podman/LXC
LOG / TRC	Journald, rsyslog, auditd, analyse des traces
IOC / PRC	Indicateurs de compromission, processus suspects, persistence
BKP	Sauvegardes, snapshots cloud
CLD	Metadata AWS, security groups, snapshots

6 – Configuration (hostaudit_cfg.py)

Chapitre 6 – Configuration (hostaudit_cfg.py)

Fichier de configuration analysé (jamais exécuté)

DISPLAY

Style de frame	: modern
Couleur	: cyan (par défaut)
Largeur	: 120
Verbose	: true

WATCH / SENTINEL / RESPONSE

Watch actif	: true
Sentinel actif	: true
Réponse automatique	: false
Actions disponibles	: isolate, notify, log
Seuil critique	: high

REPORTS

Exports automatiques	: true
Formats	: PDF, CSV, XLSX
Organisation	: par hôte / par date
Conservation (jours)	: 90

hostaudit_cfg.py

FICHIER ANALYSÉ

(JAMAIS EXÉCUTÉ)

AUDIT

Ports attendus	: 22, 80, 443, 3306, 5432
ONLY (ports)	: 22, 80, 443
SKIP (ports)	: 25, 3389
Budget (délai max)	: 300s
Budget (threads max)	: 50
Live checks Web	: true
Live checks DB	: true
Lookup CVE	: true
Lookup dépendances	: true

LOGIN & STORAGE

Utilisateur par défaut	: audit-svc
STATE_DIR	: /var/lib/hostaudit/state
DATABASE_URL	: *** (stockage chiffré)

BANNER

Afficher la bannière	: true
Fichier personnalisé	: /etc/hostaudit/banner.txt
Couleur	: cyan

ADD-ONS

Auto-install reportlab	: true
Auto-install openpyxl	: true
Vérifier les mises à jour	: true (environnement virtuel isolé)

SCHEDULER

Activé	: true
Intervalle par défaut	: 24h
Prochaine exécution	: calculée automatiquement

Chaque option de ligne de commande surcharge le paramètre correspondant du fichier de configuration.

```
Exemple : hostaudit --config hostaudit_cfg.py
--verbose false --audit-only-ports 22,443
```

Chapitre 6 – Configuration (hostaudit_cfg.py)

Le fichier de settings est **parsé** (jamais exécuté).

Il contient des sections claires :

- Display (style de frames, couleur, largeur, verbose…)
- Audit (ports attendus, ONLY/SKIP, budgets de scan, live checks web/DB, lookups CVE/dépendances…)
- Watch / Sentinel / Response
- Login & storage (utilisateur par défaut, STATE_DIR, DATABASE_URL)
- Reports (exports automatiques, organisation)
- Banner
- Add-ons (auto-install reportlab/openpyxl)
- Scheduler

Toute option ligne de commande écrase le setting correspondant.

7 – Exécution d’un audit et scoring

hostaudit

Chapitre 7 – Exécution d'un audit et scoring

command_run

1. **Chargement settings + éventuelle auto-installation.**
   Lecture du fichier de configuration et installation des dépendances si nécessaire.
2. **Authentification utilisateur (sauf --local / --no-db).**
   Vérification des credentials et obtention des droits d'accès.
3. **Création du Context.**
   Initialisation de l'environnement d'exécution et du scope d'audit.
4. **Exécution des contrôles sélectionnés (progress bar optionnelle).**
   Lancement des contrôles actifs selon le profil et les options.

   Progression 72%
5. **Calcul du score global + notes par axe + top findings.**
   Agrégation des résultats, pondération et identification des problèmes prioritaires.
6. **Stockage en base.**
   Enregistrement des résultats bruts et calculés pour traçabilité et historique.
7. **Génération des rapports demandés.**
   Création des exports dans les formats sélectionnés (HTML, JSON, PDF, CSV...).
8. **Code de sortie selon --fail-on (par défaut high).**
   Définition du statut de sortie en fonction du seuil de sévérité configuré.

MODES DE SORTIE

COMPLET (défaut)
Rapport détaillé complet avec tous les résultats, preuves, remédiations et contexte.

```
hostaudit run
[DETAILED OUTPUT]
✔ Sécurité : High | Score : 62/100
```

--short
Résumé concis ligne par ligne, idéal pour les logs et intégrations CI/CD.

```
hostaudit run --short
Score: 62/100 | High: 3 | Med: 7 | Low: 12
```

--summary
Vue synthétique par axe de sécurité avec score et tendance.

```
hostaudit run --summary
IAM: 72 | NET: 58 | SYS: 65 | DATA: 55
Score global: 62/100 (–)
```

FOCUS PAR SÉVÉRITÉ
Filtrage des résultats par niveau de sévérité (--severity high,critical).

```
hostaudit run --severity high,critical
[TOP CRITICAL/HIGH FINDINGS]
CRITICAL        4 findings
HIGH            7 findings
MEDIUM          12 findings
```

ⓘ Combinez les options pour adapter la sortie à vos besoins opérationnels.

AUDIT. SCORE. SÉCURISEZ.

Chapitre 7 – Exécution d'un audit et scoring

Flux principal (`command_run`) :

1. Chargement settings + éventuelle auto-installation.
2. Authentification utilisateur (sauf `--local` / `--no-db`).
3. Création du `Context` .
4. Exécution des contrôles sélectionnés (progress bar optionnelle).
5. Calcul du score global + notes par axe + top findings.
6. Stockage en base.
7. Génération des rapports demandés.
8. Code de sortie selon `--fail-on` (par défaut high).

Modes de sortie : complet, `--short` , `--summary` , focus par sévérité.

8 – Stockage et historique

Chapitre 8 – Stockage et historique

hostaudit stocke les données d'audit de manière durable et historisée,
et permet de suivre les évolutions dans le temps.

1. BASE PAR DÉFAUT : SQLite

SQLite

Base embarquée,
zéro dépendance

- Fichiers locaux et portables
- Idéale pour les environnements légers

STATE_DIR (par défaut)

/var/lib/hostaudit

- hostaudit/
- hostaudit.db
- backups/

La base SQLite est créée automatiquement dans STATE_DIR à la première utilisation.

2. SUPPORT DES BASES DE DONNÉES EXTERNES

Connexion via URL de connexion (JDBC-like)

```
postgresci://user:pass@host:5432/audits
```

PostgreSQL | MariaDB / MySQL | Oracle

Stockage relationnel complet, haute performance et scalable.

3. STOCKAGE DES AUDITS COMPLETS

Audits complets
Résultats, configuration, paquets, ports, services, utilisateurs, etc.

Inventaires de code (--discover)
Arborescence, métadonnées, dépendances, checksums.

Scans de secrets (--secret)
Secrets détectés, emplacements, niveau de criticité, contexte.

Chaque exécution peut être historisée avec un ID unique et un horodatage.

4. COMPARAISON INTER-AUDITS

Audit A (référence)
ID: 20250520-001 2025-05-20 10:15
ID: 20250521-002

Audit B (nouveau)
ID: 20250521-002 2025-05-21 10:30
ID: 20250521-002

→

Nouveaux éléments
- Nouveaux paquets
- Nouveaux comptes
- Fichiers ajoutés

Drifts détectés
- Modifications de configuration
- Versions modifiées
- Droits d'accès changés

Permet de mesurer l'évolution de la surface d'attaque et de valider les remédiations.

5. UTILISATEURS DE L'OUTIL

Gestion des utilisateurs locaux

Comptes sécurisés avec hash de mot de passe (PBKDF2-SHA256)

Commandes	Description
user add <user>	Ajoute un utilisateur
user list	Liste les utilisateurs
passwd <user>	Change le mot de passe
user delete <user>	Supprime un utilisateur

Les mots de passe ne sont jamais stockés en clair.

Accès aux audits et à l'historique contrôlé par authentification.

Chapitre 8 – Stockage et historique

- Base par défaut : SQLite dans `STATE_DIR` (`/var/lib/hostaudit` en root).
- Support PostgreSQL / MariaDB / Oracle via URL.
- Stockage des audits complets, inventaires de code (`--discover`), scans de secrets (`--secret`).
- Comparaison inter-audits (nouveaux éléments, drifts).
- Utilisateurs de l'outil avec hash de mot de passe (gestion `user add/list/passwd/delete`).

9 – Rapports et exports

Chapitre 9 – Rapports et exports

Exportez les résultats d'audit dans les formats adaptés à vos besoins.

Les exports peuvent être demandés à chaque audit ou sur un audit déjà stocké.
Exemple : hostaudit last --pdf

FORMATS D'EXPORT DISPONIBLES

Texte
(tableaux ASCII/Unicode)

Tableaux lisibles en ligne de commande, avec coloration automatique selon le terminal.

Exemple de sortie

```
+--------------------+--------+--------+
| HÔTE               | STATUT | RISQUES |
+--------------------+--------+--------+
| srv-app-01.example.com | CRITIQUE | 2 HIGH |
| srv-db-01.example.com  | ÉLEVÉ    | 1 HIGH |
| srv-web-01.example.com | MOYEN    | 0 HIGH |
+--------------------+--------+--------+
| Total hôtes : 3                      |
+--------------------+--------+--------+
```

PDF
(reportlab)

Rapport complet et professionnel, prêt à être partagé ou archivé.

REPORT

Rapport d'audit HostAudit

Audit ID : a1b2c3d4-5678-90ab-cdef-1234567890ab
Date : 2025-05-22 10:15:34
Hôte : srv-app-01.example.com

Résumé des risques



- Critique 2 (25%)
- Élevé 3 (37%)
- Moyen 2 (25%)
- Faible 1 (13%)

HostAudit – Sécurisez votre infrastructure

Excel
(openpyxl)

Données structurées pour analyse, tableaux croisés et reporting avancés.

Exemple de feuille

Hôte	Statut	Risque critique	Risque élevé	Risque moyen	Risque faible	Score
srv-app-01.example.com	CRITIQUE	1	0	0	92
srv-db-01.example.com	ÉLEVÉ	2	1	0	74
srv-web-01.example.com	MOYEN	0	2	1	45
srv-backup.example.com	FAIBLE	0	0	2	18

HTML
(fichier unique autonome)

Rapport autonome et interactif, consultable dans n'importe quel navigateur.

Exemple de rendu HTML

Rapport d'audit HostAudit

Hôtes	Risques critiques	Risques élevés	Score moyen
3	2	3	70/100

Détails des hôtes

- srv-app-01.example.com CRITIQUE 92
- srv-db-01.example.com ÉLEVÉ 74
- srv-web-01.example.com MOYEN 45

Voir les détails →

JSON
(complet)

Export complet et structuré pour intégration, automation et traitement machine.

Exemple de structure

```
{
 "audit_id": "a1b2c3d4-5678-90ab-cdef-1234567890ab",
 "date": "2025-05-22T10:15:34Z",
 "hosts": [
  { "hostname": "srv-app-01.example.com",
    "status": "CRITIQUE",
    "risk_score": 92, ... }
    ....
 },
 ],
 "summary": { ... }
}
```

Utilisation : hostaudit last --pdf • hostaudit export --audit-id a1b2c3d4 --format excel,json

hostaudit
AUDIT. DETECT. PROTECT.

Chapitre 9 – Rapports et exports

Quatre formats :

- **Texte** (tableaux ASCII/Unicode, couleur auto).
- **PDF** (reportlab).
- **Excel** (openpyxl).
- **HTML** (fichier unique autonome).
- **JSON** (complet).

Les exports peuvent être demandés à chaque audit ou sur un audit déjà stocké (`show last --pdf`).

10 – Mode Watch (monitoring continu)

hostaudit

SÉCURITÉ • CONTRÔLE • CONFORMITÉ

Chapitre 10 – Mode Watch (monitoring continu)

Évalue un sous-ensemble de contrôles à intervalle régulier
Intervalle par défaut : 300 secondes (configurable)

PLANIFICATION

Exécution périodique toutes les N secondes (défaut : 300 s)

Intervalle : 300 s

DÉTECTION DES DRIFTS

Compare les états attendus avec :

- Fichiers surveillés
- État runtime / système

Drift détecté

hostaudit
WATCH MODE

AUTOCORRECTION (OPTIONNELLE)

Applique automatiquement des correctifs selon la politique définie.

WATCH_AUTOFIX
Politique :
live / reload / deferred / restart

ALERTING EXTERNE

Envoie des événements JSON à une commande externe configurée.

WATCH_ALERT_COMMAND
/usr/bin/alert-handler

HEARTBEAT (OPTIONNEL)

Envoie un signal de vie à intervalle régulier pour supervision externe.

Actif toutes les 60 s

INSTALLATION SYSTEMD

Le mode Watch est conçu pour s'exécuter comme un service systemd pour une supervision continue.

hostaudit-watch.service | systemctl start hostaudit-watch | systemctl enable hostaudit-watch

Mode Watch : détection continue des dérives, réaction automatique et intégration opérationnelle.

Surveillance continue | Réaction automatique | Alerting intégré | Visibilité & fiabilité

Chapitre 10 – Mode Watch (monitoring continu)

`hostaudit watch` :

- Évalue un sous-ensemble de contrôles à intervalle régulier (défaut 300 s).
- Détecte les drifts (fichiers + état runtime).
- Peut appliquer automatiquement des correctifs selon politique (`WATCH_AUTOFIX` : live / reload / deferred / restart).
- Envoie des événements JSON à une commande externe (`WATCH_ALERT_COMMAND`).
- Heartbeat optionnel.
- Installation systemd.

11 – Sentinel (détection d’intrusion en temps réel)

hostaudit

MONITORING • SÉCURITÉ • INTÉGRITÉ

Chapitre 11 – Sentinel

détection d'intrusion en temps réel

SÉCURITÉ TEMPS RÉEL

WHEN SENTINEL = True:

DÉTECTE :
nouveaux listeners, processus depuis /tmp ou mémoire, shells sous services réseau, nouvelles clés SSH / comptes unités / setuid / modules scripts web...

INTERVALLE COURT
(défaut 10 s)

10 s

LISTE DE CONFIANCE
(SENTINEL_TRUST)

SENTINEL_TRUST

RÉPONSE AUTOMATIQUE
Les signaux peuvent être contenus automatiquement selon politique.

POLITIQUE ACTIVE

DÉTECTION EN TEMPS RÉEL

SOURCES SURVEILLÉES

- **LISTENERS RÉSEAU** — Nouveaux ports en écoute
- **PROCESSUS** — Depuis /tmp, /dev/shm ou mémoire
- **SHELLS / REVERSE SHELLS** — Sous services réseau (www-data, systemd, etc.)
- **ACCÈS & IDENTITÉS** — Clés SSH, comptes, unités
- **PRIVILÈGES & OBJETS** — setuid, capabilities, modules
- **SCRIPTS WEB** — Fichiers webshell suspects



SCAN SENTINEL
Toutes les 10 secondes

● EN COURS...

DÉTECTIONS RÉCENTES

Nouveau listener TCP 0.0.0.0:4444	10:15:23
Processus suspect /tmp/.kworker	10:15:23
Reverse shell détecté (www-data → 185.220.101.45)	10:15:24
Nouvelle clé SSH ajoutée pour root	10:15:25
Module kernel chargé : suspicious_mod	10:15:26

ANOMALIES DÉTECTÉES : 5

DERNIER SCAN : 10:15:26

ALERTE & CONTENIMENT AUTOMATIQUE

ALERTE CRITIQUE
Reverse shell détecté sur le processus PID 31247
10:15:24

ACTION AUTOMATIQUE (selon politique)

- ☑ Signal contenu
- ☑ Processus isolé
- ☑ Connexion bloquée
- ☑ Journalisation & alerte

CONTENU ✓

LISTE DE CONFIANCE (SENTINEL_TRUST)

- Ports : 22, 80, 443, 8080
- Processus : /usr/bin/nginx, /usr/sbin/sshd
- Comptes : root, admin, deploy
- Modules : wireguard, xt_state

Chapitre 11 – Sentinel (détection d'intrusion en temps réel)

Quand `SENTINEL = True` :

- Détecte : nouveaux listeners, processus depuis `/tmp` ou mémoire, shells sous services réseau, nouvelles clés SSH / comptes / unités / setuid / modules / scripts web…
- Intervalle court (défaut 10 s).
- Liste de confiance (`SENTINEL_TRUST`).
- Les signaux peuvent être contenus automatiquement selon politique.

12 – Réponse et containment (respond)

hostaudit

CYBERSECURITY • HOST INTEGRITY

Chapitre 12

Réponse et containment (respond)

Actions ciblées, réversibles et journalisées pour contenir les menaces sur les hôtes.

POLITIQUE D'OBSERVATION PUIS D'ARMEMENT

Observation pendant RESPONSE_OBSERVE_HOURS heures avant armement des actions.

DURÉE LIMITÉE OU PERMANENTE

Appliquez une durée limitée (RESPONSE_MINUTES) ou permanente (keep).

- X RESPONSE_MINUTES
- ∞ PERMANENT (keep)

TOUT EST JOURNALISÉ ET ANNULABLE

Chaque action est tracée et peut être annulée (undo) à tout moment.

ANNULABLE (undo)

ACTIONS POSSIBLES

Toujours réversibles

SUSPENDRE UN PROCESSUS (freeze)

Gèle l'exécution d'un processus sans le terminer.

BLOQUER UNE ADRESSE

Ajoute l'adresse IP ou le nom d'hôte à la liste de blocage.

VERROUILLER UN COMPTE

Empêche la connexion du compte utilisateur.

METTRE DE CÔTÉ UN FICHIER / CLÉ / SCRIPT (quarantine)

Déplace l'élément en quarantaine pour analyse.

PAUSE D'UN CONTENEUR

Met en pause le conteneur sans perte d'état.

JOURNALISATION COMPLÈTE

Toutes les actions, décisions, durées et annulations sont enregistrées de manière immuable.

RÉVERSIBILITÉ TOTALE

Toute action peut être annulée (undo) depuis l'interface ou via l'API, sans impact sur l'intégrité de l'hôte.

SÉCURISÉ • TRAÇABLE • RÉVERSIBLE

Chapitre 12 – Réponse et containment (`respond`)

Actions possibles (toujours réversibles) :

- Suspendre un processus (`freeze`)
- Bloquer une adresse
- Verrouiller un compte
- Mettre de côté un fichier / clé / script (`quarantine`)
- Pause d'un conteneur

Politique d'observation puis d'armement (`RESPONSE_OBSERVE_HOURS`).

Durée limitée (`RESPONSE_MINUTES`) ou permanente (`keep`).

Tout est journalisé et annulable (`undo`).

13 – Correctifs automatiques (--fix)

hostaudit

CYBERSECURITY INTELLIGENCE

Chapitre 13 – Correctifs automatiques (--fix)

Après un audit, hostaudit propose et applique les correctifs en toute sécurité, avec tests et rollback automatique.

1. Diff affiché

Visualisez précisément les modifications proposées avant application.

```
@@ -18,7 +18,7 @@
- PermitRootLogin yes
+ PermitRootLogin no
- PasswordAuthentication yes
+ PasswordAuthentication no
...
```

2. Backup

Un backup complet de la configuration est créé avant toute modification.

backup_20250520_142301.tar.gz

✓ Intégrité vérifiée

3. Confirmation (ou --yes)

Sécurité avant exécution. Appliquer uniquement après confirmation explicite (ou avec --yes).

--yes
Désactivé (sécurisé)
✗

Activé
(non interactif) ✓

4. Test post-application

Des tests de validation sont exécutés pour s'assurer que les correctifs fonctionnent.

Tests exécutés : 12 ✓
Échecs : 0 ✓

✓ Configuration valide

5. Rollback automatique si échec

En cas d'échec des tests, les modifications sont automatiquement annulées.

⚠ Test échoué
Rollback en cours...
Restauration terminée ✓

6. Protection forte contre la perte d'accès distant

Mécanismes de garde-fous pour préserver l'accès SSH et la connectivité critique.

- ✓ Connexions actives surveillées
- ✓ Règles SSH validées
- ✓ Délai de grâce & watchdog
- ✓ Fallback SSH sécurisé

7. Journalisation des sessions de fix

Toutes les opérations sont tracées et consultables. Historique et undo disponibles.

fixes list

Liste des sessions de correctifs appliquées.

- ID de session
- Date & heure
- Résumé des correctifs
- Statut (succès / rollback)

fixes show <id>

Affiche les détails d'une session et les modifications effectuées.

```
hostaudit fixes show fix_20250520_142301
```

fixes undo <id>

Annule une session de correctif (si possible) avec restauration du backup.

```
hostaudit fixes undo fix_20250520_142301
```

Sécurité • Fiabilité • Traçabilité • Restauration garantie

Chapitre 13 – Correctifs automatiques (`--fix`)

Après un audit, l'outil peut proposer et appliquer des correctifs :

- Diff affiché
- Backup
- Confirmation (ou `--yes`)
- Test post-application
- Rollback automatique si le test échoue
- Protection forte contre la perte d'accès distant
- Journalisation des sessions de fix (`fixes list/show/undo`)

14 – Installation, planification et administration

hostaudit

AUDIT. DETECT. PROTECT.

Chapitre 14 – Installation, planification et administration

CYBERSECURITY PRODUCT

1. AUTO-INSTALLATION EN ROOT

Installation des composants dans /usr/local/sbin/ et configuration associée.



- **hostaudit.py** — Script principal d'audit
- **settings.yaml** — Configuration générale
- **gitleaks.toml** — Règles et patterns
- **php2shell.php** — Script PHP2Shell

Exécuté en tant que root pour une installation système sécurisée.

2. INSTALLATION ET STATUT

```
# hostaudit install
✓ Vérifcation des prérequis... OK
✓ Installation des dépendances... OK
✓ Déploiement des fichiers... OK
✓ Configuration des permissions... OK
✓ Installation terminée avec succès.
```

```
# hostaudit install --status
hostaudit est installé et opérationnel.
Version : 1.4.2
Install path : /usr/local/sbin/hostaudit.py
Dernière mise à jour : 2025-05-20 10:15:23
```

3. PLANIFICATION VIA CRON (SCHEDULE INSTALL)

Utilisateur dédié

- Utilisateur : hostaudit
- Groupe : hostaudit
- Répertoires isolés
- Moins de privilèges

Planification Cron

```
# Schedule install (tous les jours à 02:00)
0 2 * * * /usr/local/sbin/hostaudit.py
--scan --report
```

Exécuté avec l'utilisateur dédié et ses credentials

Mot de passe dédié (stocké de manière sécurisée) ••••••••••

4. GESTION DES UTILISATEURS DE L'OUTIL

UTILISATEUR	RÔLE	ACCÈS	STATUT
admin	Administrateur	Tous les accès	Actif
auditor	Auditeur	Scans & rapports	Actif
viewer	Lecteur	Rapports uniquement	Actif
scanner	Automatisation	Scans uniquement	Actif

Gestion centralisée des utilisateurs et des permissions.

5. MODE LOCAL (TESTS UNIQUEMENT)



Le mode local désactive l'authentification et les contrôles de sécurité.

À n'utiliser que pour des tests locaux et non en environnement de production.

Non sécurisé – Réservé aux tests.

6. BANNER OPTIONNEL & RELANCE SOUS SUDO

Banner optionnel

```
hostaudit v1.4.2
Secure by design.
```

Personnalisable dans settings.yaml

Proposition de relance sous sudo

```
Vous n'êtes pas root.
Voulez-vous relancer avec sudo ? (o/n)
> o
```

Relance automatique avec sudo pour compléter l'installation.

Chapitre 14 – Installation, planification et administration

- Auto-installation en root → `/usr/local/sbin/hostaudit.py` + settings + gitleaks.toml + script PHP2Shell.
- Commande `install` / `install status`.
- Planification via cron (`schedule install`) avec utilisateur dédié et mot de passe.
- Gestion des utilisateurs de l'outil.
- Mode local (tests uniquement, non sécurisé).
- Banner optionnel + proposition de relance sous sudo.

Chapitre 15 – Sécurité de l'outil lui-même et limites

Points forts de sécurité :

- Allow-list stricte des commandes + politiques d'arguments.
- Lecture bornée et non-blocking.
- Fichiers de config et base de données protégés (root-only).
- Auto-installation avec vérification d'intégrité (self-test).
- Avertissements si le binaire ou le state dir sont modifiables par d'autres utilisateurs.
- Redaction des secrets dans les inventaires.

Limites :

- Principalement conçu pour Ubuntu 24.04/26.04.
- Lookups CVE/dépendances/cloud nécessitent un accès réseau (désactivés par défaut).
- Certains contrôles live (web, DB) génèrent du trafic légitime visible dans les logs.
- L'outil lui-même doit être protégé (sinon un attaquant qui le modifie obtient root au prochain run).