Audite ton serveur en 20 points

Le cahier d'exercices pour savoir où en est vraiment ta machine. Vingt vérifications, une commande à chaque fois.

audit · ton-serveur · lecture seule

```
# six parties, vingt points, un bonus
[ ] 01-04  l'accès SSH
[ ] 05-07  les comptes et les droits
[ ] 08-11  le réseau et l'exposition
[ ] 12-14  les mises à jour
[ ] 15-18  les journaux et la surveillance
[ ] 19-20  le web et les sauvegardes
[ ] bonus  tes conteneurs Docker
# score : __ / 20
```



> on ne sécurise bien que ce qu'on a d'abord regardé

Offert par Léon

zerotoroot.io

Édition octobre 2026

Debian 12 et 13 · Ubuntu 22.04 et 24.04

Avant de commencer

Un serveur exposé sur Internet se fait scanner en quelques minutes. Pas au bout de quelques mois, pas quand il devient connu : en quelques minutes. Des robots passent en continu sur toutes les adresses IP et testent les mêmes portes, partout.

La plupart des gens réagissent en empilant des tutos de durcissement. Je fais l'inverse : **d'abord je regarde**. Un audit, c'est une photo de l'état réel de la machine. Ensuite seulement on corrige, dans le bon ordre, en commençant par ce qui compte.

Ce cahier, c'est cette photo. Vingt points, regroupés en six parties : l'accès SSH, les comptes et les droits, le réseau et ce qui est exposé, les mises à jour, les journaux et la surveillance, le web et les sauvegardes. En bonus, hors score, une septième partie pour ceux qui font tourner des conteneurs Docker.

Comment l'utiliser :

- Pour chaque point, tu lances la commande, tu compares avec « ce qui est normal » et « ce qui doit t'alerter », puis tu coches.
- Tu ne corriges rien pendant l'audit. Tu notes. C'est la vue d'ensemble qui dit par où commencer.
- À la fin, tu calcules ton score avec la grille, puis la section « Lire tes résultats » t'explique ce que veut dire chaque point « à corriger », et à quel point il est grave.

Ce qu'il te faut : un serveur Debian ou Ubuntu, un accès SSH avec un utilisateur qui a les droits `sudo`, et une heure devant toi.

>  **Conseil terrain :** toutes les commandes d'audit de ce cahier sont en lecture seule. Elles regardent, elles ne modifient rien. Tu peux les lancer sur un serveur en production sans risque. Corriger, c'est une autre affaire : ça modifie la configuration, et ça ne se fait pas dans la foulée d'un audit.

Mon serveur audité :

- Nom / fournisseur : _____
- Système (résultat de `grep PRETTY /etc/os-release`) : _____
- Date de l'audit : _____

Partie 1 · L'accès SSH

SSH, c'est la porte d'entrée principale de ton serveur. C'est aussi la première que les robots essaient. Quatre vérifications.



Conseil terrain : ne te fie pas à ce que tu lis dans `/etc/ssh/sshd_config`. D'autres fichiers peuvent le surcharger (dans `/etc/ssh/sshd_config.d/`). La commande `sshd -T` affiche la configuration **réellement appliquée**, c'est elle qu'on utilise.

$ Point 1 · La connexion directe en root

```
sudo sshd -T | grep -i '^permitrootlogin'
```

Ce qui est normal : `permitrootlogin no`.

Ce qui doit t'alerter : `permitrootlogin yes`. N'importe qui peut tenter sa chance directement sur le compte le plus puissant de la machine. La valeur `prohibit-password` (clé seulement) est un entre-deux acceptable, mais si tu n'as aucune raison de te connecter en root, vise `no`.

☐ OK ☐ À corriger Ce que j'ai trouvé : _____

$ Point 2 · La connexion par mot de passe

```
sudo sshd -T | grep -iE
        '^(passwordauthentication|kbdinteractiveauthentication)'
```

Ce qui est normal : les deux lignes à `no`. On se connecte uniquement par clé.

Ce qui doit t'alerter : une des deux à `yes`. Un mot de passe se devine, se réutilise, se force à coups de dictionnaire. Une clé, non. Tant que le mot de passe est accepté, toute cette famille d'attaques reste ouverte.

☐ OK ☐ À corriger Ce que j'ai trouvé : _____

>  **Conseil terrain :** sur beaucoup de VPS Ubuntu, un fichier `/etc/ssh/sshd_config.d/50-cloud-init.conf` remet `PasswordAuthentication yes` en douce, même si tu l'as désactivé dans le fichier principal. C'est exactement pour ça qu'on audite avec `sshd -T`.

$ Point 3 · Les clés autorisées

```
sudo find /root /home -name authorized_keys -print -exec cat {} \;
ls -ld ~/.ssh ~/.ssh/authorized_keys
```

Ce qui est normal : uniquement des clés que tu reconnais (le commentaire en fin de ligne aide, par exemple `leon@portable`). Le dossier `~/.ssh` en `drwx------` (700) et le fichier en `-rw-------` (600).

Ce qui doit t'alerter : une clé que tu ne sais pas identifier, une clé d'un ancien poste ou d'un ancien prestataire, des clés dans `/root/.ssh/` alors que root est censé être fermé, ou un fichier modifiable par d'autres que toi.

☐ OK ☐ À corriger Ce que j'ai trouvé : _____

$ Point 4 · Qui frappe à ta porte

```
sudo sshd -T | grep -i '^port '
sudo journalctl -u ssh --since "24 hours ago" | grep -cE "Failed
        password|Invalid user"
```

La première commande donne le port d'écoute, la seconde compte les tentatives ratées des dernières 24 heures.

Ce qui est normal : des tentatives, il y en a toujours, c'est le bruit de fond d'Internet. Sur le port 22, quelques centaines par jour n'ont rien d'exceptionnel. Sur un port non standard, le compteur tombe en général très bas.

Ce qui doit t'alerter : des milliers de tentatives **et** la connexion par mot de passe encore active (point 2). Ou des échecs sur **ton** nom d'utilisateur depuis des adresses que tu ne connais pas.

Nombre de tentatives sur 24 h : _____ Port : _____

☐ OK ☐ À corriger



Conseil terrain : un compteur bas sur un port non standard ne veut pas dire que ton serveur est protégé. Un scan complet trouve le port en quelques minutes. Ça veut juste dire moins de bruit dans tes journaux.

Partie 2 · Les comptes et les droits

Un serveur peut avoir un SSH parfait et une porte dérobée dans ses comptes. Trois vérifications.

$ Point 5 · Les comptes qui existent vraiment

```
awk -F: '$3 == 0 {print $1}' /etc/passwd
sudo awk -F: '$2 == "" {print $1}' /etc/shadow
grep -vE '(nologin|false)$' /etc/passwd | cut -d: -f1,7
```

Dans l'ordre : les comptes qui ont les pleins pouvoirs (UID 0), les comptes sans mot de passe, les comptes qui ont un vrai shell de connexion.

Ce qui est normal : une seule ligne `root` pour la première commande. Rien pour la deuxième. Pour la troisième : `root`, ton utilisateur, et `sync:/bin/sync` (compte système historique, inoffensif).

Ce qui doit t'alerter : un deuxième compte en UID 0 (c'est une technique classique de porte dérobée), un compte sans mot de passe, ou un compte de service (`www-data`, `postgres`…) qui a un shell `/bin/bash`.

☐ OK ☐ À corriger Ce que j'ai trouvé : _____

$ Point 6 · Qui peut faire sudo

```
getent group sudo
sudo grep -rh NOPASSWD /etc/sudoers /etc/sudoers.d/
```

Ce qui est normal : dans le groupe `sudo`, uniquement les personnes qui administrent réellement la machine. Pas de `NOPASSWD`, ou alors une règle précise et voulue.

Ce qui doit t'alerter : un ancien collègue, un compte de test oublié, un compte de service dans le groupe. Ou une ligne `NOPASSWD: ALL` pour un compte humain : si ce compte est compromis, l'attaquant devient root sans rien avoir à deviner.

☐ OK ☐ À corriger Ce que j'ai trouvé : _____



Conseil terrain : sur les images cloud (Debian comme Ubuntu), l'utilisateur créé par défaut a souvent `NOPASSWD: ALL` dans `/etc/sudoers.d/90-cloud-init-users`. Ce n'est pas un piratage, c'est un réglage d'usine. Mais c'est à toi de décider si tu le gardes.

$ Point 7 · Les permissions des fichiers sensibles

```
ls -l /etc/passwd /etc/shadow /etc/sudoers
sudo find / -xdev -type f -perm -0002 -ls 2>/dev/null
```

Ce qui est normal :

- `/etc/passwd` en `-rw-r--r--` (lisible par tous, c'est prévu),
- `/etc/shadow` en `-rw-r-----`, propriétaire `root`, groupe `shadow`,
- `/etc/sudoers` en `-r--r-----`, propriétaire `root`,
- la seconde commande (fichiers modifiables par tout le monde) ne renvoie rien.

Ce qui doit t'alerter : `/etc/shadow` lisible par tous (les empreintes de mots de passe deviennent attaquables hors ligne), ou des fichiers en écriture pour tous. C'est souvent la trace d'un `chmod 777` lancé un jour « pour que ça marche ».

☐ OK ☐ À corriger Ce que j'ai trouvé : _____

Partie 3 · Le réseau et ce qui est exposé

La règle est simple : ce qui n'est pas exposé n'est pas attaquable. Quatre vérifications, de l'intérieur puis de l'extérieur.

$ Point 8 · Ce qui écoute sur le réseau

```
sudo ss -tulpn
```

Regarde la colonne de l'adresse locale :

- `127.0.0.1:...` ou `[::1]:...` : le service n'écoute que sur la machine elle-même,
- `0.0.0.0:...`, `[::]:...` ou `*:...` : le service écoute sur toutes les interfaces, donc potentiellement sur Internet.

Ce qui est normal : sur toutes les interfaces, uniquement ce que tu veux servir (SSH, et 80/443 si tu as un site). Le reste en `127.0.0.1`.

Ce qui doit t'alerter : un service que tu ne reconnais pas, ou une base de données ouverte sur toutes les interfaces (3306 MySQL/MariaDB, 5432 PostgreSQL, 6379 Redis, 27017 MongoDB, 9200 Elasticsearch). Une base ouverte sur Internet, c'est la catastrophe classique.

Port	Service	Toutes interfaces ?	Je le garde ?

☐ OK ☐ À corriger

$ Point 9 · Le pare-feu est-il vraiment actif ?

```
sudo ufw status verbose
```

Ce qui est normal : `Status: active` et `Default: deny (incoming), allow (outgoing)`. Tout est bloqué en entrée, sauf ce que tu as explicitement ouvert.

Ce qui doit t'alerter : `Status: inactive`, une politique `allow (incoming)`, ou `ufw: command not found`. Dans ce dernier cas, regarde s'il existe un autre pare-feu avec `sudo nft list ruleset`. Si la sortie est vide, ton serveur n'a pas de pare-feu.

☐ OK ☐ À corriger Ce que j'ai trouvé : _____

$ Point 10 · Chaque règle a-t-elle une raison ?

```
sudo ufw status numbered
sudo docker ps --format '{{.Names}}  {{.Ports}}' 2>/dev/null
```

Ce qui est normal : pour chaque ligne, tu sais dire ce qu'elle protège et pourquoi elle est là. SSH est soit limité (`LIMIT`), soit restreint à une adresse source.

Ce qui doit t'alerter : des règles pour des services qui n'existent plus, une base de données ouverte à `Anywhere`. Et si tu utilises Docker : un conteneur publié en `0.0.0.0:5432->5432/tcp`, par exemple (la partie 7 va plus loin).

N°	Règle	Ce qu'elle protège / pourquoi	Je la garde ?

☐ OK ☐ À corriger



Conseil terrain : Docker écrit ses propres règles de pare-feu et un port publié avec `-p 5432:5432` passe **à côté** d'UFW. Ton `ufw status` peut être parfait et ta base quand même ouverte sur Internet. C'est le piège que je vois le plus souvent.

$ Point 11 · La vue de l'extérieur

Celui-ci se lance **depuis ton poste**, pas depuis le serveur (installe `nmap` si besoin) :

```
nmap -Pn ADRESSE_IP_DE_TON_SERVEUR
```

Par défaut, `nmap` teste les 1 000 ports les plus courants. Ajoute `-p-` pour tester les 65 535 (plus long).

Ce qui est normal : tu retrouves exactement les ports que tu as décidé d'ouvrir, rien d'autre.

Ce qui doit t'alerter : un port ouvert qui n'apparaît pas dans tes règles de pare-feu (souvent Docker, voir point 10), ou un service que tu pensais réservé à la machine.

Ports vus de l'extérieur : _____

☐ OK ☐ À corriger



Conseil terrain : scanne uniquement des machines qui t'appartiennent. Regarde aussi si ton hébergeur propose un pare-feu dans son interface (Hetzner, Scaleway, OVH… en ont un) et s'il est actif. Si oui, ce que voit `nmap` est le résultat des deux couches.

Partie 4 · Les mises à jour

La grande majorité des compromissions passent par une faille connue, déjà corrigée, mais pas encore installée. Trois vérifications.

$ Point 12 · Les mises à jour en attente

```
sudo apt update
apt list --upgradable
```

Ce qui est normal : une liste vide, ou quelques paquets publiés depuis hier.

Ce qui doit t'alerter : des dizaines de paquets en attente, et surtout des lignes qui contiennent `-security` (par exemple `bookworm-security` sur Debian 12, `noble-security` sur Ubuntu 24.04). Ce sont des failles corrigées que ton serveur n'a pas encore reçues.

Nombre de paquets en attente : _____ dont sécurité : _____

☐ OK ☐ À corriger

$ Point 13 · Les correctifs de sécurité automatiques

```
cat /etc/apt/apt.conf.d/20auto-upgrades
sudo tail -n 5 /var/log/unattended-upgrades/unattended-upgrades.log
```

Ce qui est normal : le fichier contient ces deux lignes, chacune à `"1"` :

```
APT::Periodic::Update-Package-Lists "1";
APT::Periodic::Unattended-Upgrade "1";
```

Et le journal montre une activité récente (moins de quelques jours).

Ce qui doit t'alerter : fichier absent, valeurs à `"0"`, ou dernier passage qui date de plusieurs semaines.

☐ OK ☐ À corriger Ce que j'ai trouvé : _____



Conseil terrain : ce qu'on attend ici, ce sont les **correctifs de sécurité** appliqués sans intervention, pas les montées de version majeures. Un correctif de sécurité, je le veux cette nuit sans y penser. Un passage de Debian 12 à 13 ne doit jamais arriver tout seul.

$ Point 14 · Un système encore suivi, et vraiment redémarré

```
grep PRETTY /etc/os-release
uptime
uname -r
ls /boot/vmlinuz-*
ls /var/run/reboot-required 2>/dev/null
```

Ce qui est normal : une version encore maintenue (Debian 12 ou 13, Ubuntu 22.04 ou 24.04 LTS). Le noyau qui tourne (`uname -r`) est le plus récent de ceux installés dans `/boot` . Pas de fichier `reboot-required` .

Ce qui doit t'alerter : une version en fin de vie (Debian 11 ou plus ancien, une Ubuntu non LTS ancienne, Ubuntu 20.04 ou plus ancien sans abonnement de support étendu), ou un noyau installé depuis des semaines mais jamais chargé faute de redémarrage. Une mise à jour du noyau ne protège qu'après le redémarrage.

☐ OK ☐ À corriger Ce que j'ai trouvé : _____



Conseil terrain : un `uptime` de 400 jours n'est pas un trophée. C'est souvent 400 jours de correctifs noyau installés mais jamais actifs.

Partie 5 · Les journaux et la surveillance

Un serveur qu'on ne surveille pas, c'est un serveur dont on apprend la compromission par quelqu'un d'autre. Quatre vérifications.

$ Point 15 · Fail2ban fait-il son travail ?

```
sudo fail2ban-client status sshd
```

Ce qui est normal : la prison `sshd` existe, `Total failed` augmente avec le temps, et des adresses apparaissent dans `Banned IP list`.

Ce qui doit t'alerter : `command not found` (fail2ban n'est pas installé), une erreur indiquant que la prison `sshd` n'existe pas, ou des compteurs à zéro alors que le point 4 montre des centaines de tentatives. Dans ce dernier cas, fail2ban tourne mais ne lit pas le bon journal.

☐ OK ☐ À corriger Ce que j'ai trouvé : _____



Conseil terrain : depuis Debian 12, le fichier `/var/log/auth.log` n'existe plus par défaut (tout passe par le journal systemd). Un fail2ban configuré « à l'ancienne » cherche ce fichier, ne le trouve pas, et ne bannit personne. Le symptôme : le service tourne, mais les compteurs restent à zéro.

$ Point 16 · Les connexions réussies

```
sudo journalctl -u ssh --since "7 days ago" | grep "Accepted"
```

Ce qui est normal : des lignes `Accepted publickey for ton_utilisateur from ...`, depuis des adresses que tu reconnais (ta box, ton bureau, ton VPN), à des heures cohérentes avec ton activité.

Ce qui doit t'alerter : une ligne `Accepted password` alors que tu pensais le mot de passe désactivé, un compte que tu n'utilises pas, une adresse inconnue, une connexion à 3 h du matin un jour où tu dormais.

Adresses que je reconnais : _____

☐ OK ☐ À corriger

$ Point 17 · Des journaux conservés, et de la place pour les écrire

```
ls -d /var/log/journal
journalctl --list-boots | head -5
journalctl --disk-usage
df -h /
```

Ce qui est normal : le dossier `/var/log/journal` existe (les journaux survivent au redémarrage), plusieurs démarrages apparaissent dans la liste si la machine a déjà redémarré, et le disque est rempli à moins de 80 %.

Ce qui doit t'alerter : journaux non conservés (après un incident suivi d'un redémarrage, tu n'as plus rien à analyser), ou un disque à plus de 90 %. Un disque plein, c'est des journaux qui ne s'écrivent plus et des services qui tombent.

Occupation du disque : _____ %

☐ OK ☐ À corriger

$ Point 18 · Les couches de défense avancées (AppArmor et auditd)

```
sudo aa-status | head -5
sudo auditctl -l
```

Ce qui est normal : `apparmor module is loaded.` et un certain nombre de profils `in enforce mode`. Pour auditd, une liste de règles qui surveillent tes fichiers sensibles (configuration SSH, comptes, sudoers).

Ce qui doit t'alerter : AppArmor non chargé. Pour auditd, `command not found` ou `No rules` signifie simplement que cette couche n'est pas en place.

☐ OK ☐ À corriger Ce que j'ai trouvé : _____



Conseil terrain : c’est la couche qu’on ajoute en dernier, une fois les bases solides. Les permissions Unix disent **qui** peut toucher un fichier. AppArmor dit ce que **ce programme** a le droit de faire, même lancé en root. Et auditd garde la trace de qui a modifié quoi, au niveau du noyau. Note : sur certains VPS à base de conteneurs, ces deux outils ne sont pas disponibles.

Partie 6 · Le web et les sauvegardes

Deux derniers points. Le premier ne concerne que les serveurs qui publient un site ou une application web. Le second concerne tout le monde, sans exception.

$ Point 19 · HTTPS, certificat et vieux protocoles

Si ton serveur ne publie aucun service web, coche « non concerné » et passe au point 20.

Depuis ton poste, en remplaçant `ton-domaine.fr` :

```
curl -sI http://ton-domaine.fr | head -3
echo | openssl s_client -connect ton-domaine.fr:443 -servername ton-
       domaine.fr 2>/dev/null | openssl x509 -noout -enddate
nmap --script ssl-enum-ciphers -p 443 ton-domaine.fr
```

Ce qui est normal :

- la première commande renvoie une redirection `301` (ou `308`) vers `https://`,
- la date de fin du certificat est dans le futur, et il se renouvelle tout seul,
- `nmap` ne liste que `TLSv1.2` et `TLSv1.3`.

Ce qui doit t'alerter : le site répond en HTTP clair sans rediriger, un certificat qui expire dans quelques jours sans renouvellement automatique, ou `TLSv1.0` / `TLSv1.1` encore acceptés.

☐ OK ☐ À corriger ☐ Non concerné

$ Point 20 · Les sauvegardes, et la restauration

Ici, pas de commande universelle : tout dépend de ton outil. Commence par chercher ce qui tourne :

```
systemctl list-timers --all
sudo crontab -l
crontab -l
```

Puis réponds honnêtement :

- Mes sauvegardes sont automatiques : ☐ oui ☐ non
- Elles partent **hors** de ce serveur (autre machine, autre hébergeur) : ☐ oui ☐ non
- Date de la dernière sauvegarde réussie : _____
- J'ai déjà restauré un fichier depuis une sauvegarde, pour de vrai : ☐ oui ☐ non

Ce qui est normal : quatre « oui » et une date de moins de 48 heures.

Ce qui doit t'alerter : pas de sauvegarde, une sauvegarde stockée sur le même disque, ou une sauvegarde jamais testée. Une sauvegarde jamais restaurée n'est pas une sauvegarde, c'est un espoir.

☐ OK ☐ À corriger

>  **Conseil terrain :** l'instantané (snapshot) proposé par ton hébergeur est pratique avant une grosse manipulation, mais il vit chez le même fournisseur que ton serveur. Si le compte est bloqué ou le centre de données en panne, il part avec. Au point 20, un instantané seul ne compte pas comme une copie hors du serveur.

Partie 7 · Tes conteneurs Docker (bonus)

Si tu n'utilises pas Docker, passe directement à la page suivante. Pour vérifier : si `sudo docker ps` répond `command not found`, Docker n'est pas installé.

Cette partie est un bonus : elle ne compte pas dans le score sur 20. Mais Docker change les règles du jeu sur plusieurs points du cahier, à commencer par le pare-feu (point 10). Six vérifications, les mêmes sur Debian et sur Ubuntu.

$ D1 · Les ports publiés

```
sudo docker ps --format 'table {{.Names}}\t{{.Ports}}'
```

Ce qui est normal : les services internes publiés sur `127.0.0.1` (par exemple `127.0.0.1:8080->80/tcp`), ou pas publiés du tout. Seul ton proxy inverse (Caddy, Nginx, Traefik) apparaît en `0.0.0.0:80` et `0.0.0.0:443`.

Ce qui doit t'alerter : une base de données ou une interface d'administration en `0.0.0.0:...` ou `[::]:...`, par exemple `0.0.0.0:5432->5432/tcp`. Docker écrit ses propres règles de pare-feu : ce port est ouvert sur Internet, quoi qu'en dise `ufw status`.

☐ OK ☐ À corriger Ce que j'ai trouvé : _____

$ D2 · Qui est dans le groupe docker

```
getent group docker
```

Ce qui est normal : la ligne se termine par ton utilisateur (celui qui administre la machine), ou par rien.

Ce qui doit t'alerter : n'importe quel autre compte. Être dans le groupe `docker`, c'est pouvoir lancer un conteneur qui monte tout le disque de la machine : c'est être root, sans mot de passe et sans passer par `sudo`.

☐ OK ☐ À corriger Ce que j'ai trouvé : _____

$ D3 · Le socket Docker monté dans un conteneur

```
sudo docker ps -q | xargs -r sudo docker inspect --format '{{.Name}}:
        {{range .Mounts}}{{.Source}} {{end}}' | grep docker.sock
```

La commande liste les conteneurs qui ont accès à `/var/run/docker.sock`, la prise de commande de Docker.

Ce qui est normal : rien, ou uniquement un outil que tu as installé en connaissance de cause (un proxy inverse qui lit les étiquettes des conteneurs, un tableau de bord d'administration).

Ce qui doit t'alerter : une application ordinaire (un site, un service web) qui a le socket. Qui contrôle ce socket contrôle Docker, donc la machine entière. Et le monter en lecture seule (`:ro`) n'y change rien : on peut toujours envoyer des ordres à Docker à travers.

☐ OK ☐ À corriger Ce que j'ai trouvé : _____

$ D4 · Les conteneurs privilégiés ou en root

```
sudo docker ps -q | xargs -r sudo docker inspect --format '{{.Name}}
        privileged={{.HostConfig.Privileged}}  user={{.Config.User}}
        réseau={{.HostConfig.NetworkMode}}'
```

Ce qui est normal : `privileged=false` partout. Idéalement un utilisateur renseigné après `user=` (un nom ou un numéro). Un réseau propre au projet (par exemple `monappli_default`).

Ce qui doit t'alerter : `privileged=true`, qui donne au conteneur un accès quasi complet à la machine. Un `user=` vide veut dire que le processus tourne en root dans le conteneur : c'est très courant, mais une faille dans l'application donne alors root dans le conteneur, un pas de moins vers la machine. Et `réseau=host`, qui fait écouter le conteneur directement sur les interfaces de la machine.

☐ OK ☐ À corriger Ce que j'ai trouvé : _____

$ D5 · Les images jamais mises à jour

```
sudo docker ps --format '{{.Names}}  {{.Image}}  {{.RunningFor}}'
sudo docker images --format '{{.Repository}}:{{.Tag}}  {{.CreatedSince}}'
```

La première commande donne l'image de chaque conteneur et depuis quand il tourne. La seconde, l'âge de chaque image présente sur la machine.

Ce qui est normal : des images construites il y a quelques semaines, idéalement avec un numéro de version précis (`postgres:16`, `nginx:1.27`) plutôt que `latest`.

Ce qui doit t'alerter : une image `latest` construite il y a plusieurs mois, ou un conteneur qui tourne depuis un an. `latest` ne se met pas à jour tout seul : c'est la version du jour où tu l'as téléchargée, avec toutes les failles découvertes depuis.

☐ OK ☐ À corriger Ce que j'ai trouvé : _____

$ D6 · Les secrets en clair

D'abord, retrouve où sont tes fichiers compose (si la commande n'existe pas, ils sont souvent dans `/srv`, `/opt` ou ton dossier personnel) :

```
sudo docker compose ls
```

Puis cherche les fichiers qui contiennent des secrets, et les `.env` lisibles par tous :

```
sudo grep -rliE 'password|passwd|secret|token|api_key' --include='*.yml' --
        include='*.yaml' --include='.env' /srv /opt /home 2>/dev/null
sudo find /srv /opt /home -name '.env' -perm -o=r -ls 2>/dev/null
```

La première commande n'affiche que des noms de fichiers, jamais les valeurs. Adapte les dossiers à ce que t'a montré `docker compose ls`.

Ce qui est normal : les secrets rangés dans un fichier `.env` à part, lisible par son seul propriétaire (`-rw-------`), et jamais versionné dans Git. La seconde commande ne renvoie rien.

Ce qui doit t'alerter : un mot de passe écrit directement dans un `docker-compose.yml`, surtout si ce fichier est dans un dépôt Git poussé sur une forge. Ou un `.env` lisible par tous les comptes de la machine.

☐ OK ☐ À corriger Ce que j'ai trouvé : _____

Partie 7 : ___ / 6 (hors score)

Ton serveur est-il prêt ?

Compte 1 point par case « OK » (et par « non concerné » au point 19). La partie 7 (Docker) ne compte pas dans le score.

Partie	Points	Mon score
1 · L'accès SSH	1 à 4	___ / 4
2 · Les comptes et les droits	5 à 7	___ / 3
3 · Le réseau et l'exposition	8 à 11	___ / 4
4 · Les mises à jour	12 à 14	___ / 3
5 · Les journaux et la surveillance	15 à 18	___ / 4
6 · Le web et les sauvegardes	19 et 20	___ / 2
Total		**___ / 20**

Comment lire ton score :

- **18 à 20 : solide.** Ton serveur a de vraies bases. Le travail maintenant, c'est de rendre tout ça reproductible (et de refaire cet audit tous les trois mois).
- **13 à 17 : de bonnes bases, des trous.** Rien d'alarmant si les points rouges ci-dessous sont au vert. Commence par les points graves (voir « Lire tes résultats »).
- **8 à 12 : vulnérable.** Ton serveur tient surtout parce que personne ne s'y est encore intéressé de près. Prévois une session de correction cette semaine.
- **0 à 7 : porte ouverte.** Si le serveur héberge des données qui comptent, traite-le comme une priorité, quitte à le reconstruire proprement.

Les points rouges, quel que soit ton score. Ces cinq points pèsent plus lourd que les autres. Si l'un d'eux est « à corriger », c'est par lui que tu commences :

☐ Point 2 · connexion par mot de passe

☐ Point 8 · base de données ou service inconnu exposé

☐ Point 9 · pas de pare-feu actif

☐ Point 12 · correctifs de sécurité en attente

☐ Point 20 · pas de sauvegarde hors du serveur

Si tu utilises Docker, ajoute à cette liste tout point D1, D2 ou D3 « à corriger ».

La contre-épreuve automatique. Une fois ton audit à la main terminé, lance un audit automatisé :

```
sudo apt install lynis
sudo lynis audit system
```

Lynis te donne un `Hardening index` (sur 100) et une liste de suggestions. Note ton score maintenant, et relance-le après tes corrections : c'est la progression qui compte, pas le chiffre de départ.

- Score Lynis avant corrections : _____
- Score Lynis après corrections : _____

Mes trois corrections prioritaires :

1. Priorité 1 : _____
2. Priorité 2 : _____
3. Priorité 3 : _____

> et maintenant, corriger

Tu as ta liste de problèmes. « Sécuriser son serveur » te montre comment corriger chacun, pas à pas.

Ce cahier te dit où ça coince. La formation te montre comment corriger : les commandes, pourquoi chaque réglage compte, dans quel ordre les appliquer, et comment ne pas te couper l'accès en route. C'est le Module 2 de De Zéro à Root, disponible seul.

- **12 vidéos** pas à pas : SSH, pare-feu (politique par défaut et règles avancées), fail2ban, mises à jour automatiques, AppArmor, auditd, HTTPS.
- **Un lab Docker jetable**, pour tout essayer et tout casser avant de toucher à ta vraie machine.
- **Un livret PDF** avec la checklist de durcissement et les commandes clés.

29,90 € · accès à vie · garantie 14 jours

> bastion.zerotoroot.io

Lire tes résultats

Tu as ton score et tes cases cochées. Cette section t'aide à lire chaque résultat : ce qu'il veut dire, et à quel point il est sérieux. Trois niveaux :

- **Grave** : à traiter en premier, cette semaine. C'est une porte ouverte, ou le signe que quelqu'un est peut-être déjà passé.
- **À surveiller** : pas une urgence, mais un affaiblissement réel. À traiter une fois les points graves réglés.
- **Mineur** : du confort ou de l'hygiène. Ça compte sur la durée, pas aujourd'hui.

Ici, tu ne trouveras pas les corrections. Corriger un accès SSH ou un pare-feu sur une machine distante, c'est justement là qu'on se coupe l'accès : ça mérite mieux qu'une ligne de commande copiée en bas d'un PDF.

$ Partie 1 · L'accès SSH

- **Point 1, root en SSH.** `permitrootlogin yes` : **grave**. Le compte le plus puissant de la machine est offert aux essais de tous les robots. `prohibit-password` : **mineur**, si tu sais pourquoi tu en as besoin.
- **Point 2, mot de passe accepté.** Une des deux lignes à `yes` : **grave**, c'est un point rouge. Tant que le mot de passe passe, chaque tentative du point 4 a une chance de réussir.
- **Point 3, clés autorisées.** Une clé que personne ne sait expliquer : **grave**, et c'est plus qu'un réglage. Quelqu'un a peut-être un accès. Avant de toucher à quoi que ce soit, regarde le point 16. Une clé d'un ancien poste ou d'un ancien prestataire : **à surveiller**. Des permissions trop larges sur `~/.ssh` : **à surveiller**.
- **Point 4, qui frappe à ta porte.** Des centaines de tentatives seules : **mineur**, c'est le bruit de fond d'Internet. Des milliers de tentatives avec le mot de passe encore accepté : **grave**, c'est le point 2 qui pèse. Des échecs sur ton propre nom d'utilisateur depuis des adresses inconnues : **à surveiller**, quelqu'un te vise toi, pas une liste au hasard.

La correction pas à pas de ces quatre points est dans « Sécuriser son serveur » : bastion.zerotoroot.io.

$ Partie 2 · Les comptes et les droits

- **Point 5, les comptes.** Un deuxième compte en UID 0 que tu n'as pas créé : **grave**. Ce n'est pas un oubli de configuration, c'est le signe classique d'une intrusion. Ne le supprime pas dans la précipitation : regarde d'abord le point 16 et note ce que tu trouves. Un compte sans mot de passe : **grave**. Un compte de service avec un vrai shell : **à surveiller**.
- **Point 6, sudo.** Un ancien collègue, un compte de test ou un compte de service dans le groupe `sudo` : **grave**, c'est un accès root que plus personne ne surveille. Un `NOPASSWD: ALL` sur ton propre compte : **à surveiller**, c'est souvent le réglage d'usine des images cloud.
- **Point 7, les permissions.** `/etc/shadow` lisible par tous : **grave**, les empreintes de mots de passe peuvent être copiées et attaquées hors ligne. Des fichiers modifiables par tout le monde : **à surveiller**, et **grave** si c'est un script lancé par root ou un fichier de `/etc`.

Les comptes et les permissions sont la base du Module 1 de De Zéro à Root : boot.zerotoroot.io.

$ Partie 3 · Le réseau et l'exposition

- **Point 8, ce qui écoute.** Une base de données sur toutes les interfaces : **grave**, c'est un point rouge et la catastrophe la plus fréquente. Un service que tu ne reconnais pas : **grave** tant que tu ne sais pas ce que c'est. Un service connu mais inutile : **mineur**, c'est une surface d'attaque gratuite.
- **Point 9, le pare-feu.** Pas de pare-feu, ou une politique `allow (incoming)` : **grave**, point rouge. Ta seule protection est alors que rien d'inutile n'écoute (point 8).
- **Point 10, les règles.** Une base ouverte à `Anywhere`, ou un conteneur publié en `0.0.0.0` : **grave**. Des règles pour des services disparus : **mineur**. SSH ouvert à tous sans limitation : **mineur** si le mot de passe est refusé (point 2), **à surveiller** sinon.
- **Point 11, la vue de l'extérieur.** Le moindre écart entre ce que voit `nmap` et ce que tu croyais ouvert : **grave** par principe. Ça veut dire que l'image que tu as de ton serveur est fausse. Le plus souvent, c'est Docker (voir la partie 7).

La correction pas à pas de ces quatre points est dans « Sécuriser son serveur » : bastion.zerotoroot.io.

$ Partie 4 · Les mises à jour

- **Point 12, en attente.** Des paquets `-security` en attente : **grave**, point rouge, surtout s'ils attendent depuis des semaines. Ce sont des failles publiques, avec leur mode d'emploi en ligne. Des mises à jour ordinaires en attente : **mineur**.
- **Point 13, correctifs automatiques.** Désactivés : **à surveiller**. **Grave** si le point 12 montre déjà du retard : la preuve est faite que personne ne passe à la main.
- **Point 14, version et redémarrage.** Une version en fin de vie : **grave**. Elle ne reçoit plus aucun correctif, le point 12 ne pourra jamais être au vert. Un noyau installé mais pas chargé : **à surveiller**, la faille corrigée est toujours là en mémoire.

La correction pas à pas de ces trois points est dans « Sécuriser son serveur » : bastion.zerotoroot.io.

$ Partie 5 · Les journaux et la surveillance

- **Point 15, fail2ban.** Absent : **à surveiller** si le mot de passe est refusé, **grave** s'il est accepté. Installé mais compteurs à zéro alors que le point 4 montre des tentatives : **à surveiller**, c'est le pire des cas, tu crois être protégé et tu ne l'es pas.
- **Point 16, les connexions réussies.** Une ligne `Accepted password` alors que tu pensais le mot de passe désactivé : **grave**, ta configuration n'est pas celle que tu crois. Une connexion que tu ne peux pas expliquer : **grave**, c'est le résultat le plus sérieux de tout le cahier. Ne modifie rien dans la précipitation : copie les lignes du journal (date, compte, adresse) et garde-les hors du serveur.
- **Point 17, journaux et disque.** Journaux non conservés : **à surveiller**, après un incident suivi d'un redémarrage, tu n'auras rien à analyser. Disque entre 80 et 90 % : **à surveiller**. Au-delà de 90 % : **grave**, les journaux et les services vont tomber.
- **Point 18, AppArmor et auditd.** AppArmor non chargé : **à surveiller**. auditd absent : **mineur**, c'est une couche avancée qui vient après les bases. Sur un VPS à base de conteneurs, ces deux outils peuvent être indisponibles : non applicable.

La correction pas à pas des points 15 et 18 est dans « Sécuriser son serveur » : bastion.zerotoroot.io.

$ Partie 6 · Le web et les sauvegardes

- **Point 19, HTTPS.** Un site qui répond en HTTP clair sans rediriger, ou un certificat qui expire sans renouvellement automatique : **grave**, données en clair ou site bientôt inaccessible. `TLSv1.0` ou `TLSv1.1` encore acceptés : **à surveiller**.

- **Point 20, les sauvegardes.** Pas de sauvegarde, ou seulement sur le même serveur : **grave**, point rouge. Le jour où le disque, le compte ou la machine tombe, tout part avec. Une sauvegarde jamais restaurée : **à surveiller**, tu ne sais pas si elle fonctionne. Une dernière sauvegarde de plus de 48 heures : **à surveiller**.

La correction pas à pas du point 19 est dans « Sécuriser son serveur » : bastion.zerotoroot.io. Les sauvegardes (point 20) sont traitées dans De Zéro à Root, Module 7 : boot.zerotoroot.io.

$ Partie 7 · Tes conteneurs Docker (bonus)

- **D1, ports publiés.** Une base de données ou un service interne publié en `0.0.0.0` : **grave**, il est joignable depuis Internet même si ton pare-feu dit le contraire. Seul ton proxy inverse en `0.0.0.0:80` et `0.0.0.0:443` : normal.
- **D2, le groupe docker.** Un compte autre que l'administrateur dans le groupe : **grave**, ce compte est root sur la machine, sans mot de passe.
- **D3, le socket Docker.** Monté dans un conteneur d'application : **grave**, si l'application est compromise, la machine l'est aussi. Monté dans un outil d'administration que tu as choisi (proxy inverse, tableau de bord) : **à surveiller**, c'est ton conteneur le plus sensible.
- **D4, privilèges.** `privileged=true` : **grave**. Un conteneur qui tourne en root (champ `user` vide) : **à surveiller**, c'est très courant mais c'est un risque. Réseau `host` : **à surveiller**.
- **D5, images.** Une image `latest` créée il y a plusieurs mois : **à surveiller**, et **grave** si le service est exposé sur Internet.
- **D6, secrets.** Des mots de passe dans un fichier compose versionné, et surtout poussé sur une forge : **grave**, considère-les comme connus. Un `.env` lisible par tous : **à surveiller**.

Docker, ses pièges et sa sécurité sont traités dans De Zéro à Root, Module 5 : boot.zerotoroot.io.

La suite

Tu as maintenant la photo de ton serveur, ton score, et pour chaque point ce qu'il veut dire et à quel point il est sérieux. C'est déjà beaucoup : la plupart des serveurs en ligne n'ont jamais eu droit à cet examen.

Mais un audit dit **quoi** corriger. Il ne remplace pas le fait de l'avoir fait de tes mains, en comprenant ce que chaque couche protège et pourquoi l'ordre compte. C'est là que la plupart des gens se coupent l'accès à leur propre serveur, un soir, en activant un pare-feu une ligne trop tôt.

La suite logique : « Sécuriser son serveur ». C'est le Module 2 de De Zéro à Root, disponible seul. 12 vidéos pas à pas sur l'essentiel de ce cahier (SSH, pare-feu, fail2ban, mises à jour automatiques, AppArmor, auditd, HTTPS), un lab Docker jetable où tu peux tout casser sans risque avant de toucher à ta vraie machine, et un livret PDF avec la checklist de durcissement et les commandes clés. 29,90 €, accès à vie, garantie 14 jours.

`> bastion.zerotoroot.io`

Ensuite, si tu veux tout le parcours, ce module fait partie de **De Zéro à Root**, la formation complète : des bases de Linux jusqu'à l'automatisation avec Ansible, Docker, l'intégration continue et la supervision. Durcir une fois, c'est bien. Le rendre reproductible sur toutes tes machines, c'est le métier.

`> boot.zerotoroot.io`

Dans les deux cas, garde ce cahier. Refais l'audit dans trois mois, et compare les scores.

`>_ On se retrouve de l'autre côté. · Léon`