hermian security

Guide de réponse à incident Linux

Quoi taper, et dans quel ordre, lorsqu'une machine Linux a été compromise. À lancer en root.

Mohamed Gning · Mouhamed M. Sabaly

01 **Détecter**
02 **Trier**
03 **Préserver**
04 **Contenir**
05 **Éradiquer**
06 **Rétablir**



Six règles avant de toucher à quoi que ce soit

01 Isolez la machine du réseau en gardant votre accès, avant de nettoyer quoi que ce soit (p. 20).

02 Ne redémarrez pas et n'éteignez pas : la mémoire est la meilleure preuve.

03 Ne faites pas confiance aux outils de la machine. Recoupez avec `/proc` et un binaire apporté (p. 4).

04 Sortez les preuves sur une clé USB ou un disque externe, jamais sur le disque suspect (p. 18).

05 Notez chaque commande avec l'heure UTC.

06 Ne tapez aucun nouveau secret sur la machine : changez-les depuis un poste sain (p. 21).

Démarrer un journal

- **Monter la clé USB ou le disque externe qui recevra les preuves**

```
mkdir -p /mnt/ir && mount /dev/sdX1 /mnt/ir
```

- **Enregistrer toute la session dessus**

```
script -q -a /mnt/ir/ir-$(hostname)-$(date -u +%Y%m%dT%H%MZ).log
```

- **Afficher l'heure UTC dans chaque invite**

```
export TZ=UTC PS1='[\D{%F - %T}] \u@\h:\w# '
```

- **Identifier la machine**

```
date -u; hostnamectl; uptime; uname -a
```

- **Ultime recours, sans disque : le collecteur se connecte à la machine suspecte (à lancer sur le collecteur)**

```
ssh root@suspect 'ps auxf' > ps.txt
```

Jamais l'inverse : une connexion lancée depuis la machine compromise utilise son client `ssh`, qui peut être truqué, et l'obligerait à détenir une clé vers le collecteur. Même dans ce sens, l'attaquant voit la connexion et la commande passe par ses outils. Avant d'en arriver là, cherchez une clé USB, un disque externe, la console du fournisseur ou un instantané du volume. Utilisez un collecteur dédié à cet incident, et calculez l'empreinte des fichiers dès leur arrivée.

Quand les binaires du système ne sont plus fiables, utilisez les vôtres. Copiez un `busybox` statique dans `/mnt/ir` et lancez `/mnt/ir/busybox ps`, `ls` ou `netstat`.

Les 15 premières minutes, dans cet ordre

1	Peut-on faire confiance aux outils ?	p. 4
2	Qui est ou était connecté ?	p. 5
3	Qu'est-ce qui tourne, et qui l'a lancé ?	p. 7
4	Avec qui la machine communique-t-elle ?	p. 9
5	Comment l'intrus survit-il à un redémarrage ?	p. 11
6	Qu'est-ce qui a changé sur le disque et dans les journaux ?	p. 14

Si quelque chose sort du lot, arrêtez le triage. Préservez (p. 18), puis contenez (p. 20).

Peut-on faire confiance aux outils ?

- **Sur un système sain, /etc/ld.so.preload n'existe pas, ou il est vide**

```
ls -la /etc/ld.so.preload
cat /etc/ld.so.preload
```

- **S'il n'est pas vide : le sauvegarder comme preuve, avec les bibliothèques qu'il charge et leur paquet** — RHEL / Fedora `rpm -qf`

```
cp -a /etc/ld.so.preload /mnt/ir/ && sha256sum /etc/ld.so.preload > /mnt/ir/ld.so.preload.sha256
for l in $(cat /etc/ld.so.preload); do cp -a "$l" /mnt/ir/; sha256sum "$l"; dpkg -S "$l" || echo "unowned: $l"; done
```

Les usages légitimes sont rares : Raspberry Pi OS y charge `libarmmem`, et quelques outils de journalisation (Snoopy) ou agents de sécurité s'y inscrivent. Une bibliothèque qu'aucun paquet ne possède, ou cachée dans `/usr/lib` sous un nom anodin, est un rootkit jusqu'à preuve du contraire. Si les outils du système ne montrent pas le fichier, relisez-le avec votre `busybox` statique.

Videz-le seulement une fois les preuves sorties et la machine contenue, avec `truncate -s 0 /etc/ld.so.preload`. Les processus déjà lancés gardent la bibliothèque en mémoire : redémarrez-les, ou mieux, reconstruisez la machine.

- **LD_PRELOAD défini dans votre propre shell**

```
env | grep -i '^ld_'
```

- **Noyau « tainted » : toute valeur autre que 0 signale un module non signé ou externe**

```
cat /proc/sys/kernel/tainted
```

- **Modules chargés, vus de deux façons : les noms doivent correspondre**

```
lsmod
cat /proc/modules
```

- **Fichiers de paquets modifiés depuis l'installation** — RHEL / Fedora `rpm -Va`

```
dpkg --verify | grep -v ' c /'
```

- **Processus cachés à ps mais présents dans /proc**

```
ps -eo pid= | sort -n > /mnt/ir/ps.pids
ls /proc | grep -E '^[0-9]+$' | sort -n > /mnt/ir/proc.pids
diff /mnt/ir/ps.pids /mnt/ir/proc.pids
```

Quelques lignes de `diff` sont normales, des processus naissent et meurent entre les deux commandes. Un PID présent dans `/proc` et absent de `ps`, en revanche, mérite qu'on s'y arrête.

Qui est ou était connecté ?

- **Qui est sur la machine en ce moment**

```
w
who -a
```

- **Historique des connexions avec l'IP source, puis les échecs**

```
last -Faiw | head -40
lastb -Faiw | head -20
```

- **Authentifications SSH des dernières 48 heures** — RHEL / Fedora `journalctl -u sshd`

```
journalctl -u ssh --since "48 hours ago" | grep -E 'Accepted|Invalid'
```

- **Connexions réussies dans le journal d'authentification** — RHEL / Fedora `/var/log/secure`

```
grep -E 'Accepted|session opened' /var/log/auth.log
```

- **Tous les comptes avec l'UID 0, puis les membres du groupe root (GID 0)**

```
awk -F: '$3==0 {print $1}' /etc/passwd
awk -F: '$4==0 {print $1}' /etc/passwd; getent group root
```

- **Comptes et groupes modifiés : comparer avec la version précédente**

```
diff /etc/passwd- /etc/passwd
diff /etc/group- /etc/group
diff /etc/shadow- /etc/shadow
```

Les fichiers terminés par - sont la version précédente, gardée par les outils de gestion des comptes : ils ne montrent que le dernier changement. Une sauvegarde d'avant l'incident reste la meilleure référence.

- **Qui peut devenir root** — RHEL / Fedora `getent group wheel`

```
getent group sudo adm docker lxd
grep -v '^#' /etc/sudoers /etc/sudoers.d/* | grep .
```

- **Chaque fichier authorized_keys, et son contenu**

```
find / -xdev -name 'authorized_keys*' -ls -exec cat {} \;
```

SIGNAUX D'ALERTE un second compte UID 0, une clé sans commentaire, une connexion depuis un pays où vous n'opérez pas, un `Accepted password` sur un serveur censé n'accepter que des clés.

Qu'est-ce qui tourne, et qui l'a lancé ?

- **L'arbre complet des processus, avec leurs heures de démarrage**

```
ps -eo pid,ppid,user,lstart,etime,args --forest
```

- **Binaires supprimés, ou lancés depuis un répertoire temporaire**

```
ls -l /proc/*/exe 2>/dev/null | grep -E 'deleted|/tmp|/dev/shm|/var/tmp'
```

- **Processus qui travaillent depuis un répertoire temporaire**

```
ls -l /proc/*/cwd 2>/dev/null | grep -E '/tmp|/dev/shm|/var/tmp'
```

- **Faux threads noyau : un [nom] dont le parent n'est pas kthreadd (PID 2)**

```
ps -eo pid,ppid,args | awk '$3 ~ /^\[/ && $2 != 2 && $1 > 2'
```

Examiner un processus de près

- Où il se trouve

```
P=4121
ls -l /proc/$P/exe /proc/$P/cwd
```

- Comment il a été lancé, et avec quel environnement

```
tr '\0' ' ' < /proc/$P/cmdline; echo
tr '\0' '\n' < /proc/$P/environ
```

- Ce qu'il a ouvert

```
ls -l /proc/$P/fd
lsof -nP -p $P
```

- Sauvegarder le binaire, même s'il a été supprimé du disque

```
cp /proc/$P/exe /mnt/ir/$P.exe
sha256sum /mnt/ir/$P.exe
```

Ce qui n'est pas normal

- Un serveur web ou une base de données avec un shell comme enfant : `nginx → sh`, `php-fpm → bash`, `postgres → sh`.
- Un programme lancé depuis `/tmp`, `/dev/shm`, `/var/tmp` ou un répertoire caché.
- Un lien `exe` qui se termine par `(deleted)` : le fichier a été supprimé après le lancement.
- `curl` ou `wget` envoyé dans `sh`, ou `base64 -d` dans une ligne de commande.
- Un nom de processus qui ne correspond pas à son `exe`, comme `[kworker/0:2]` qui pointe vers `/tmp/.x/kw`.
- Un CPU occupé en continu par un nom inconnu : les mineurs sont la charge la plus courante sur les serveurs exposés.

Avec qui la machine communique-t-elle ?

- **Chaque socket, avec le processus qui la détient**

```
ss -tunap
lsof -nP -i
```

- **Ports en écoute**

```
ss -ltnup
```

- **Connexions établies**

```
ss -tnp state established
```

- **Interfaces, routes et voisins**

```
ip -br a; ip route; ip neigh
```

- **Résolution de noms : quelque chose que vous n'avez pas configuré ?**

```
cat /etc/resolv.conf /etc/hosts
```

- **Règles de pare-feu que vous n'avez pas écrites**

```
nft list ruleset
iptables-save
```

SIGNAUX D'ALERTE un port en écoute que vous ne savez pas nommer, des sorties vers les ports 1080, 3333, 4444, 5555 ou 14444 (proxys, mineurs, shells), des connexions détenues par un processus dans `/tmp`, une entrée `/etc/hosts` pour votre miroir de paquets.

Conteneurs

- **Tous les conteneurs, y compris arrêtés**

```
docker ps -a --no-trunc
```

- **Ceux qui tournent en mode privilégié**

```
docker ps -q | xargs docker inspect -f '{{.Name}} privileged={{.HostConfig.Privileged}}'
```

- **Les images, avec leurs empreintes**

```
docker images --digests
```

- **L'API Docker est-elle ouverte sur le réseau ?**

```
ss -ltnp | grep -E ':2375|:2376'
```

- **Nœuds containerd et Kubernetes**

```
crictl ps -a
```

Un conteneur privilégié, ou qui monte `/`, donne root sur l'hôte. Le port 2375 ouvert sert souvent aux mineurs.

Comment l'intrus survit-il à un redémarrage ?

- **Cron, pour chaque utilisateur**

```
for u in $(cut -d: -f1 /etc/passwd); do crontab -l -u "$u" 2>/dev/null | sed "s/^/$u: /"; done
```

- **Cron système et tâches at** — RHEL / Fedora `/var/spool/cron`

```
cat /etc/crontab
ls -la /etc/cron.* /var/spool/cron/crontabs
atq
```

- Services et minuteurs activés

```
systemctl list-unit-files --state=enabled
systemctl list-timers --all
```

- Fichiers d'unités modifiés ces 14 derniers jours

```
find /etc/systemd /lib/systemd /usr/lib/systemd -type f -mtime -14 -ls
```

- Unités systemd par utilisateur

```
find /root /home -path '*/.config/systemd/user/*' -ls
```

- Scripts de shell et de connexion

```
ls -la /etc/profile.d
cat /etc/environment /etc/bash.bashrc
cat /root/.bashrc /root/.profile /home/*/.bashrc /home/*/.profile
```

- Bibliothèques préchargées et chemins de bibliothèques

```
cat /etc/ld.so.preload
ls -la /etc/ld.so.conf.d
```

- Lignes PAM qui lancent un programme

```
grep -rE 'pam_exec|pam_script' /etc/pam.d
```

- **Modules PAM qu'aucun paquet ne possède**

```
for f in /lib/x86_64-linux-gnu/security/*.so; do
  dpkg -S "$f" >/dev/null 2>&1 || dpkg -S "/usr$f" >/dev/null 2>&1 || echo "unowned: $f"; done
```

- **Règles udev qui lancent un programme**

```
grep -r 'RUN' /etc/udev/rules.d
```

- **Crochets de démarrage et de connexion**

```
cat /etc/rc.local
ls -la /etc/update-motd.d
```

- **Crochets apt**

```
grep -r 'Invoke' /etc/apt/apt.conf.d
```

- **Chaîne de démarrage : noyau, initramfs et GRUB, puis l'état de Secure Boot**

RHEL / Fedora `rpm -V kernel-core grub2-common`

```
ls -la --time-style=full-iso /boot /boot/grub
dpkg -V $(dpkg -S /boot/vmlinuz-$(uname -r) | cut -d: -f1) grub-common
mokutil --sb-state
```

Un initramfs régénéré à la main, une entrée GRUB inconnue ou un module dans `/lib/modules` qu'aucun paquet ne possède méritent un examen. Un implant dans le micrologiciel (UEFI) survit à une réinstallation : si vous le soupçonnez, faites appel à un spécialiste.

- **Répertoires SSH**

```
ls -la /root/.ssh /home/*/.ssh
```

Où les attaquants s'installent

`authorized_keys`	Une nouvelle clé dans le fichier de root ou d'un compte de service
`cron`	Une ligne qui retélécharge la charge malveillante
`unité systemd`	Un service au nom anodin, activé à la main
`ld.so.preload`	Masque fichiers et processus à tous les outils
`profils de shell`	Tourne à chaque connexion, souvent masqué
`PAM`	Enregistre les mots de passe ou accepte un mot de passe maître

Qu'est-ce qui a changé sur le disque ?

- **Nouveaux fichiers setuid et setgid**

```
find / -xdev -type f -perm /6000 -mtime -30 -ls
```

- **Capacités de fichiers**

```
getcap -r / 2>/dev/null
```

- **Exécutables et fichiers cachés dans les répertoires temporaires**

```
find /tmp /var/tmp /dev/shm -type f -perm -u+x -ls
find / -xdev -name '.*' -path '*/tmp/*' -ls
```

La seconde commande remonte aussi des fichiers légitimes : `/tmp/.X11-unix`, `.ICE-unix`, fichiers d'éditeurs et verrous. Regardez d'abord les exécutables, et les noms qui imitent un fichier système ou ne contiennent que des points et des espaces.

- **Fichiers système modifiés cette semaine**

```
find /etc /usr/bin /usr/sbin /usr/lib -xdev -type f -mtime -7 -ls
```

- **Intégrité des paquets** — RHEL / Fedora `rpm -Va`

```
dpkg --verify
debsums -c 2>/dev/null
```

- **Paquets installés ou mis à jour récemment : plus rapide pour repérer une porte dérobée fraîche** — RHEL / Fedora `rpm -qa --last | head -40`

```
zgrep -h ' install \| upgrade ' /var/log/dpkg.log* | sort | tail -40
```

- **Date de naissance d'un binaire clé**

```
stat /usr/sbin/sshd
```

`touch` peut changer la date de modification, mais pas la date de changement (`ctime`) ni, sur ext4 et xfs, la date de naissance. Commencez par celles-là.

Que disent les journaux ?

- Les démarrages, puis une fenêtre de temps

```
journalctl --list-boots
journalctl --since "2026-11-10 08:00" --until "2026-11-10 12:00"
```

- Avertissements et erreurs de la dernière journée

```
journalctl -p warning --since "24 hours ago"
```

- Fichiers du journal abîmés (ne prouve pas qu'ils sont complets)

```
journalctl --verify
```

- Fichiers de logs vides ou tronqués

```
ls -la --time-style=full-iso /var/log
```

- Requêtes web qui méritent d'être lues

```
zgrep -hE 'POST|wget|curl|base64|/bin/sh' /var/log/nginx/access.log*
```

- Historique du shell : un lien vers /dev/null est un signal d'alerte

```
ls -la /root/.bash_history
cat /root/.bash_history /home/*/.bash_history
```

- **Commandes exécutées, si auditd est installé**

```
ausearch -m EXECVE -ts today
```

SIGNAUX D'ALERTE un trou dans les journaux, un fichier de logs vide ou recréé récemment, un historique lié à `/dev/null`. C'est souvent le premier signe. `journalctl --verify` ne prouve rien : un journal effacé puis recréé proprement passe la vérification. Notez quand le trou commence et quand il finit.

Recherches rapides

- **Configurations de mineurs**

```
grep -rlE 'stratum\+tcp|xmrig|minerd|c3pool|nanopool' /etc /root /home /tmp /var/tmp /dev/shm 2>/dev/null
```

- **Shells qui tiennent une connexion TCP**

```
ss -tnp | grep -E '"(bash|sh|dash|nc|ncat|socat|python3?|perl|php)"'
```

- **Entrées cron qui téléchargent quelque chose**

```
grep -rE 'curl|wget' /etc/cron* /var/spool/cron 2>/dev/null
```

- **Fichiers authorized_keys modifiés ce mois-ci**

```
find / -xdev -name authorized_keys -mtime -30 -ls
```

Reconstituer une chronologie à partir des dates de fichiers

- **Chaque fichier modifié dans une fenêtre, du plus ancien au plus récent (touch ne peut pas fixer ctime)**

```
find / -xdev -newerct '2026-11-09 00:00' ! -newerct '2026-11-11 00:00' -type f \
     -printf '%C+ %u %m %p\n' 2>/dev/null | sort > /mnt/ir/timeline.txt
```

- **La lire sans le bruit des paquets**

```
grep -vE ' /var/(cache|lib/apt|lib/dpkg)/' /mnt/ir/timeline.txt | less
```

Commencez la fenêtre un jour avant le premier signe. Les changements qui arrivent ensemble (une charge, une entrée cron, une clé, un binaire) tombent souvent à la même minute.

Le plus volatil d'abord

1	Mémoire	perdue au redémarrage
2	Processus, sockets, fichiers ouverts	perdus à leur arrêt
3	Disque	change à chaque écriture
4	Journaux distants et sauvegardes	en général sûrs

- **1. La mémoire, avec AVML (binaire statique, github.com/microsoft/avml)**

```
/mnt/ir/avml /mnt/ir/mem.lime
```

- **2. L'état en direct**

```
mkdir /mnt/ir/state && cd /mnt/ir/state
ps -eo pid,ppid,user,lstart,args --forest > ps.txt
ss -tunap > ss.txt; lsof -nP > lsof.txt; ip a > ip.txt
```

- **Ou tout le triage d'un coup, avec UAC (github.com/tclahr/uac)**

```
/mnt/ir/uac/uac -p ir_triage /mnt/ir
```

- **3. Le disque, sur un disque externe plus grand que lui**

```
dd if=/dev/vda of=/mnt/ir/vda.img bs=4M conv=noerror,sync status=progress
```

- **Ultime recours, sans disque externe : lancé depuis le collecteur**

```
ssh root@suspect 'dd if=/dev/vda bs=4M' | dd of=vda.img bs=4M status=progress
```

Copié pendant que le système tourne, le disque donne une image incohérente : fichiers ouverts, données encore en cache. Si vous pouvez arrêter la machine, faites-le après la capture mémoire, de préférence en coupant l'alimentation (un arrêt propre exécute des scripts que l'attaquant a pu modifier), puis imagez le disque depuis un système de secours. Sinon, notez dans le journal que l'image a été prise à chaud.

- **4. Calculer l'empreinte de tout ce que vous avez collecté**

```
cd /mnt/ir && find . -type f -exec sha256sum {} + > SHA256SUMS
```

Dans le cloud, commencez par un instantané du volume. C'est plus rapide que `dd`, et l'invité n'est pas touché.

Couper l'accès à l'attaquant, garder le vôtre

- Avant d'isoler : relever les connexions en cours, ce sont des preuves

```
ss -tunap > /mnt/ir/connexions-avant-isolement.txt
```

- N'autoriser que votre IP en SSH, tout le reste est coupé, y compris les connexions déjà ouvertes (nftables)

```
ADMIN=198.51.100.10
nft add table inet ir
nft add chain inet ir in  '{ type filter hook input priority -10; policy drop; }'
nft add chain inet ir out '{ type filter hook output priority -10; policy drop; }'
nft add rule inet ir in  iif lo accept
nft add rule inet ir in  ip saddr $ADMIN tcp dport 22 accept
nft add rule inet ir out oif lo accept
nft add rule inet ir out ip daddr $ADMIN tcp sport 22 accept
```

- Tout annuler

```
nft delete table inet ir
```

- Geler un processus au lieu de le tuer : sa mémoire reste lisible

```
kill -STOP 4121
```

- **Verrouiller et faire expirer un compte**

```
usermod -L -e 1 mallory
```

Aucune règle n'accepte les connexions « déjà établies » en général : celles de l'attaquant, ouvertes avant l'isolement, sont coupées aussi. Avant d'appliquer, gardez une deuxième session ouverte depuis votre IP pour vérifier que vous gardez la main. Dans le cloud, modifiez plutôt le groupe de sécurité ou isolez l'instance depuis la console.

Nettoyer pour de bon

01 Trouvez d'abord le point d'entrée. Si la porte reste ouverte, vous serez de retour ici la semaine prochaine.

02 Réinstallez depuis une image saine plutôt que de nettoyer. Un rootkit qui a survécu au nettoyage ne se voit pas.

03 Changez tous les secrets que la machine pouvait lire, depuis une machine saine : clés SSH, jetons d'API, mots de passe de bases de données, clés cloud, fichiers `.env`.

04 Restaurez les données d'avant le premier signe, et vérifiez-les avant de remettre la machine en service.

05 Surveillez 30 jours. Mêmes IP, mêmes domaines, mêmes comptes.

Avec Hermian sur la machine

- **Les alertes, de la plus récente à la plus ancienne**

```
sudo hermian alerts -s HIGH
```

- **L'alerte complète, avec sa chaîne de processus**

```
sudo hermian show HER-2026-1110-003
```

- **Une archive de preuves pour cette alerte**

```
sudo hermian collect HER-2026-1110-003
```

- **Les signes d'une compromission antérieure à l'installation de Hermian**

```
sudo hermian audit
```

- **Couper les sessions SSH d'une adresse, puis la bloquer**

```
sudo hermian disconnect 203.0.113.9
sudo hermian block 203.0.113.9
```

- **Isoler la machine pendant une heure, en gardant l'accès d'administration**

```
sudo hermian isolate --for 60
sudo hermian unisolate
```

Les alertes gardent déjà la chaîne de processus et les connexions, donc une partie des étapes 2 et 3 est faite.

Tout consigner

- La chronologie en UTC : premier signe, détection, confinement, rétablissement.
- Les indicateurs : IP, domaines, empreintes de fichiers, chemins, noms d'utilisateurs.
- Le point d'entrée, et comment vous l'avez confirmé.
- Les actions menées, par qui et quand.
- Le périmètre : machines, comptes, clés et données que l'attaquant pouvait atteindre.
- Ce qui aurait permis de le voir plus tôt, et à qui revient ce chantier.

hermian
security

Quelque chose d'inexpliqué ?

Une alerte que vous ne comprenez pas, un serveur qui se comporte bizarrement, une trace que vous n'arrivez pas à expliquer : écrivez-nous. On répond sous 48 heures, en français ou en anglais.

contact@hermian.me

Version 1.0, novembre 2026 CC BY 4.0