PortailSIO

NOUVELLE FORMULE

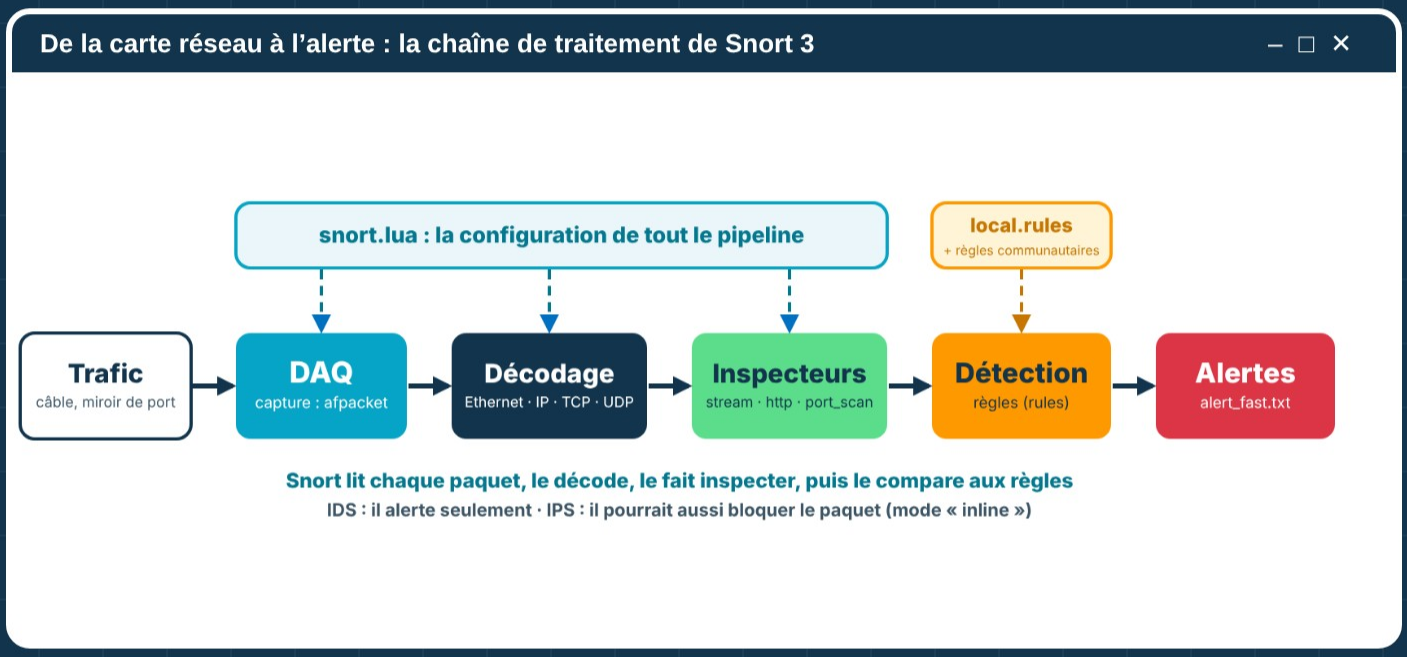
TUTORIEL

SNORT 3 · IDS · DEBIAN 13

Détecter les intrusions avec SNORT



NIVEAU
Expert



1 Comprendre un IDS et préparer le serveur Debian 13

2 Installer les dépendances et compiler LibDAQ

3 Compiler et installer Snort 3

4 Configurer Snort et écrire ses premières règles

5 Détecter : ping, scan de ports et règles communautaires

6 Exploiter : service, journaux et mises à jour

CC BY-NC

Utilisation commerciale interdite

© PortailSIO – 11/2026

Ce tutoriel en ligne

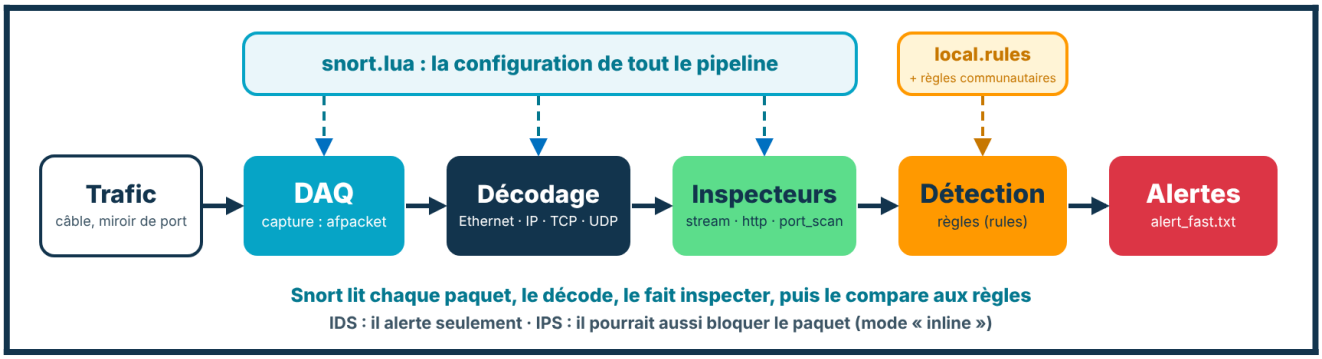
Scannez le QR code. Tous les tutoriels :

portail.sio-ndlp.fr/tutoriels.php

Ce tutoriel a été réalisé avec l'hyperviseur © Proxmox VE. Il peut être réalisé dans un environnement de virtualisation personnel à l'aide des logiciels © VirtualBox ou © VMware Player. Il **suppose que vous disposez d'une machine virtuelle Debian 13** (installation minimale avec serveur SSH) et d'une seconde machine Linux pour générer du trafic de test.

Snort est le système de **détection d'intrusion** libre le plus répandu. Développé par Cisco Talos (licence GPL v2), il **lit le trafic réseau**, le **compare à des règles** décrivant des attaques connues et **écrit une alerte** à chaque correspondance. **Snort 3**, la version utilisée ici (3.12), est une réécriture complète : configuration en **Lua** (fichier `snort.lua`), architecture en **modules** (les « inspecteurs »), traitement multitâche. Snort 3 ne se trouve pas dans les paquets de Debian : on le **compile** à partir des sources, ce qui est aussi l'occasion de comprendre ses composants.

Le trafic traverse une **chaîne de traitement** unique : la carte réseau fournit les paquets à **DAQ** (la couche de capture), qui les remet à Snort ; ceux-ci sont **décodés**, passés aux **inspecteurs** (suivi des connexions TCP, protocole HTTP, détection de balayage de ports…), puis comparés aux **règles** par le moteur de détection. Le fichier `snort.lua` configure toute la chaîne :



IDS ou IPS ?

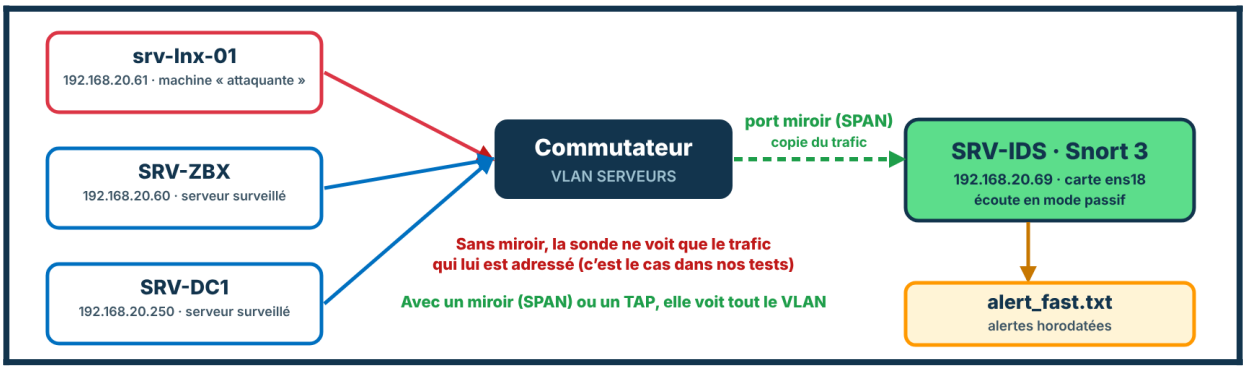
	IDS (détection)	IPS (prévention)
Position	**Hors du flux** : il reçoit une **copie** du trafic (port miroir, TAP)	**En coupure** : le trafic le **traverse** (deux interfaces)
Action	Il **alerte** et journalise	Il alerte **et peut bloquer** le paquet (`drop`, `reject`)
Risque	Aucun pour le trafic : une panne de la sonde ne coupe rien	Un faux positif **bloque du trafic légitime** ; une panne coupe la liaison
Mode DAQ	Passif (`afpacket` sur une interface)	Inline (`afpacket` sur deux interfaces, option `-Q`)

Ce tutoriel met en place un **IDS** : c'est le bon point de départ, sans risque, et celui qui permet de **régler les règles** avant d'envisager de bloquer quoi que ce soit.

REMARQUE IMPORTANTE

N'analysez que le réseau dont vous avez la responsabilité. L'interception de correspondances échangées par voie électronique est une infraction pénale (article 226-15 du Code pénal), et la charte informatique de l'établissement encadre l'installation de sondes. Pour un TP, demandez l'accord de l'administrateur du réseau et travaillez sur les machines virtuelles du laboratoire. Les scans de ports de ce tutoriel ne visent que **SRV-IDS**, votre propre machine.

Où placer la sonde ? Une carte réseau ne reçoit normalement que le trafic qui **lui est adressé**. Pour qu'une sonde voie le trafic des autres machines, le commutateur doit lui envoyer une **copie** : c'est le **port miroir** (SPAN) ou un **TAP**. Dans notre laboratoire, les tests visent la sonde elle-même, ce qui ne demande aucune configuration réseau :



Ce que l'on met en place

- **Snort 3** compilé et installé dans `/usr/local/snort`, avec sa bibliothèque de capture **LibDAQ**
- **Une configuration** : réseau protégé (`HOME_NET`), règles locales, journal des alertes
- **Des détections** testées : un ping, une tentative SSH et un **scan de ports**, puis les **règles communautaires**
- **L'exploitation** : service systemd sans privilèges `root`, rotation des journaux, mise à jour automatique des règles

Dans le cadre de ce tutoriel, nous utilisons les machines suivantes (VLAN SERVEURS, réseau 192.168.20.0 /24) :

Machine	Rôle	Système	Adresse IP
SRV-IDS	Sonde Snort 3 (nom `srv-ids`), carte `ens18`	Debian 13	192.168.20.69
srv-lnx-01	Machine de test : envoie un ping et un scan de ports	Debian 13	192.168.20.61

Pour **SRV-IDS**, prévoyez **2 vCPU, 4 Go de mémoire et 20 Go de disque** : la compilation de Snort est gourmande en mémoire (avec moins de 4 Go, compilez avec `make -j1`). Ces valeurs, comme les adresses, les noms et les noms de carte, sont à adapter à votre environnement. Les commandes sont exécutées avec un compte autorisé à utiliser `sudo`.

ÉTAPE 1 SUR 6

01 Comprendre un IDS et préparer le serveur Debian 13

Une sonde a besoin d'un système à jour, d'un nom, d'une adresse fixe et d'une **heure exacte** : les alertes sont horodatées et servent à recouper des événements entre machines. Ouvrez une session (console Proxmox ou SSH) sur **SRV-IDS** puis effectuez les manipulations suivantes :

1. Mettez le système à jour et installez l'outil de réglage des cartes réseau : `sudo apt update && sudo apt full-upgrade -y && sudo apt install -y ethtool`

2 Donnez son nom au serveur, réglez le fuseau horaire et déclarez le nom dans le fichier `hosts` :

TERMINAL

```
sudo hostnamectl set-hostname srv-ids
echo "192.168.20.69  srv-ids.sio.local  srv-ids" | sudo tee -a /etc/hosts
sudo timedatectl set-timezone Europe/Paris
```

3 Donnez au serveur une **adresse IP fixe** (adaptez le nom de la carte, ici `ens18`, avec `ip -br address`) : éditez `/etc/network/interfaces` puis redémarrez le réseau avec `sudo systemctl restart networking`. La ligne `post-up` désactive deux fonctions de la carte, expliquées juste après :

/etc/network/interfaces

```
auto ens18
iface ens18 inet static
    address 192.168.20.69/24
    gateway 192.168.20.254
    post-up /usr/sbin/ethtool -K ens18 gro off lro off
```

4 Indiquez les **DNS du domaine** dans `/etc/resolv.conf` (`search sio.local`, `nameserver 192.168.20.250`, `nameserver 192.168.20.251`)

5 Vérifiez le résultat : nom, adresse IP, fonctions de la carte et ressources disponibles :

TERMINAL

```
hostnamectl | grep -E 'Static|Operating'
ip -br address show ens18
ethtool -k ens18 | grep -E 'generic-receive|large-receive'
nproc && free -h | sed -n 1,2p
```

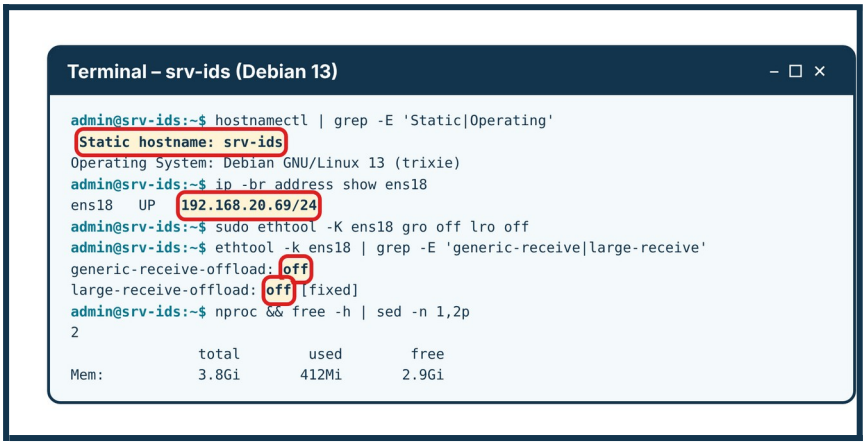


Figure 1 – Contrôle de la préparation de la sonde

À SAVOIR

GRO et **LRO** sont des fonctions de la carte qui **regroupent plusieurs paquets en un seul** avant de les remettre au système. Snort verrait alors des paquets plus gros que ceux qui ont réellement circulé : les règles et les calculs de taille seraient faussés. On les désactive sur toute carte d'une sonde. Selon le pilote, `large-receive-offload` peut afficher « off [fixed] » : c'est normal.

✔ **POINT DE CONTRÔLE**

- `hostnamectl` affiche le nom **srv-ids** et Debian GNU/Linux **13**
- L'adresse IP **192.168.20.69** est fixe sur `ens18`
- **generic-receive-offload** et **large-receive-offload** sont **off**
- La machine dispose d'au moins **2 vCPU** et de **4 Go** de mémoire

02 ÉTAPE 2 SUR 6 Installer les dépendances et compiler LibDAQ

Snort s'appuie sur plusieurs bibliothèques. Nous installons les **outils de compilation** et les **fichiers de développement** (paquets `-dev`) depuis Debian, puis nous compilons **LibDAQ**, la couche de capture de paquets, qui n'est pas non plus empaquetée par Debian.

A – Installer les dépendances

1 Installez les paquets nécessaires à la compilation (une seule commande, la barre oblique inverse continue la ligne) :

TERMINAL

```
sudo apt install -y build-essential cmake git pkg-config autoconf libtool flex bison \
  libpcap-dev libpcre2-dev libdumbnet-dev libluajit-5.1-dev libhwloc-dev zlib1g-dev \
  libssl-dev libunwind-dev liblzma-dev uuid-dev libsafec-dev libnghttp2-dev
```

Paquet	Rôle
`build-essential`, `cmake`, `autoconf`, `libtool`, `pkg-config`, `git`	Compilateurs et outils de construction ; `git` récupère les sources
`flex`, `bison`	Générateurs d'analyseurs, nécessaires à la lecture des règles
`libpcap-dev`, `libdumbnet-dev`	Capture de paquets et manipulation des adresses réseau (la bibliothèque « dnet » s'appelle **dumbnet** dans Debian)
`libpcre2-dev`, `libluajit-5.1-dev`	Expressions régulières des règles et interpréteur **Lua** du fichier `snort.lua`
`libssl-dev`, `zlib1g-dev`, `liblzma-dev`, `libnghttp2-dev`	Chiffrement, décompression et analyse HTTP/2
`libhwloc-dev`, `libunwind-dev`, `uuid-dev`, `libsafec-dev`	Gestion des cœurs du processeur, diagnostic d'erreurs, identifiants et fonctions mémoire sécurisées (`libsafec-dev` est facultatif)

À SAVOIR

Hyperscan (`libhyperscan-dev`) est un moteur de recherche de motifs optimisé qui accélère nettement la détection sur un réseau chargé. Il est facultatif : le laboratoire s'en passe, et Snort l'utilisera automatiquement s'il est installé avant la compilation.

B – Compiler et installer LibDAQ

Les sources sont téléchargées dans un dossier de travail du compte utilisateur (pas besoin de `root` pour compiler, seulement pour installer). LibDAQ s'installe dans `/usr/local`, que Debian parcourt par défaut :

1 Téléchargez LibDAQ et placez-vous sur la version **3.0.28**, la version minimale exigée par Snort 3.12 :

TERMINAL

```
mkdir -p ~/snort-src && cd ~/snort-src
git clone https://github.com/snort3/libdaq.git
cd libdaq && git checkout v3.0.28
```

2 Compilez et installez, puis actualisez le cache des bibliothèques :

TERMINAL

```
./bootstrap
./configure
make -j$(nproc)
sudo make install
sudo ldconfig
```

3 Vérifiez l'installation : les modules de capture, la bibliothèque et la version :

TERMINAL

```
ls /usr/local/lib/daq
ldconfig -p | grep libdaq
pkg-config --modversion libdaq
```

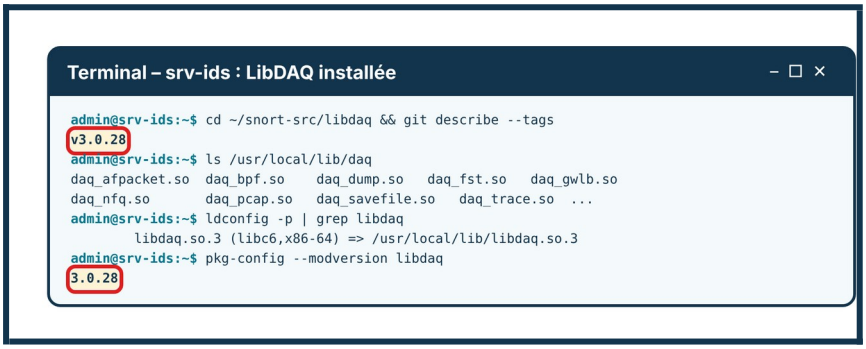


Figure 2 – LibDAQ installée

À SAVOIR

Un **module DAQ** est un « pilote » de capture : **afpacket** lit directement les paquets de la carte (c'est le mode des sondes Linux), **pcap** passe par libpcap et sait aussi **relire une capture** enregistrée, **nfq** s'interface avec le pare-feu `netfilter` pour le mode IPS.

✔ **POINT DE CONTRÔLE**

- Le dossier `/usr/local/lib/daq` contient **daq_afpacket.so** et **daq_pcap.so**
- `pkg-config --modversion libdaq` affiche **3.0.28**
- Vous savez expliquer le rôle de **DAQ** dans la chaîne de traitement

03 ÉTAPE 3 SUR 6 Compiler et installer Snort 3

A – Télécharger les sources

1 Téléchargez Snort 3 et placez-vous sur la version **3.12.4.0** (une version figée garantit des résultats identiques d'une machine à l'autre) :

TERMINAL

```
cd ~/snort-src
git clone https://github.com/snort3/snort3.git
cd snort3 && git checkout 3.12.4.0
```

B – Configurer la compilation

1 Lancez le script de configuration, qui prépare la compilation dans un dossier `build`. L'option `--prefix` choisit le dossier d'installation (c'est aussi la valeur par défaut) :

TERMINAL

```
./configure_cmake.sh --prefix=/usr/local/snort
```

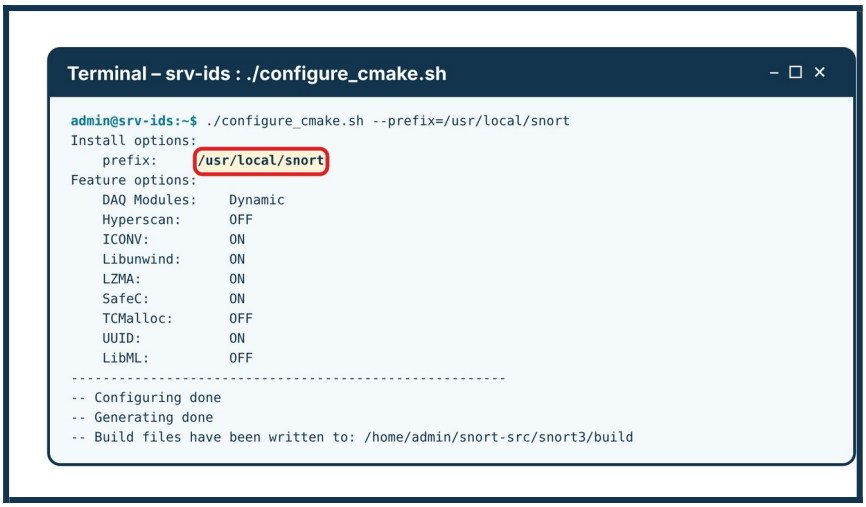


Figure 3 – Configuration de la compilation

À SAVOIR

Le résumé liste les fonctions détectées : `Hyperscan : OFF` signifie que le moteur optimisé est absent (acceptable ici), `LibML : OFF` désactive une analyse expérimentale par apprentissage automatique. Si le script s'arrête sur « Could not find DAQ », c'est que `ldconfig` n'a pas été lancé ou que LibDAQ n'est pas installée : reprenez l'étape 2.

C – Compiler puis installer

La compilation de Snort est **longue** : prévoyez de **15 à 40 minutes** selon les ressources de la machine. Lancez-la dans une session que vous pouvez laisser ouverte (ou dans `tmux`) :

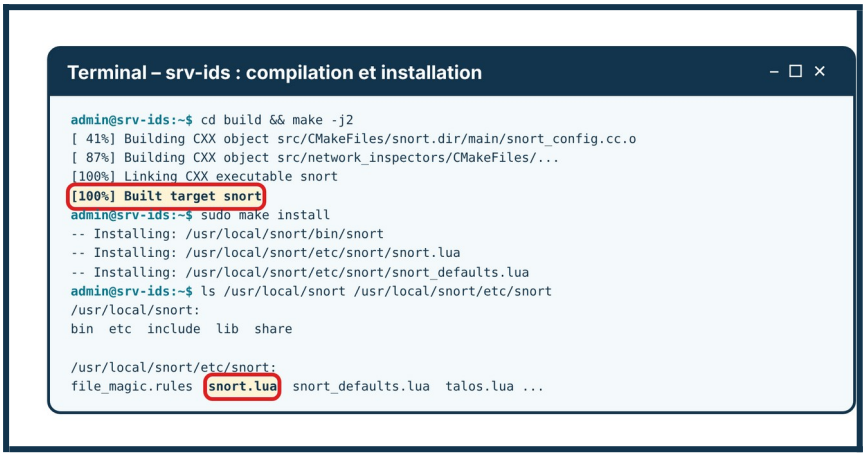
1 Compilez avec autant de tâches que de cœurs (voir l'avertissement ci-dessous), puis installez :

TERMINAL

```
cd build
make -j$(nproc)
sudo make install
```

REMARQUE IMPORTANTE

Si la compilation s'interrompt avec un message **Killed** ou « cc1plus: fatal error », le système a manqué de **mémoire** et le noyau a tué le compilateur. Relancez avec moins de tâches parallèles (`make -j1`) ou ajoutez de la mémoire ou du swap à la machine. Relancer `make` reprend là où il s'est arrêté.



À SAVOIR

L'installation place le programme dans `/usr/local/snort/bin` et les **fichiers de configuration modèles** dans `/usr/local/snort/etc/snort` : `snort.lua` (la configuration principale), `snort_defaults.lua` (valeurs par défaut) et `talos.lua`.

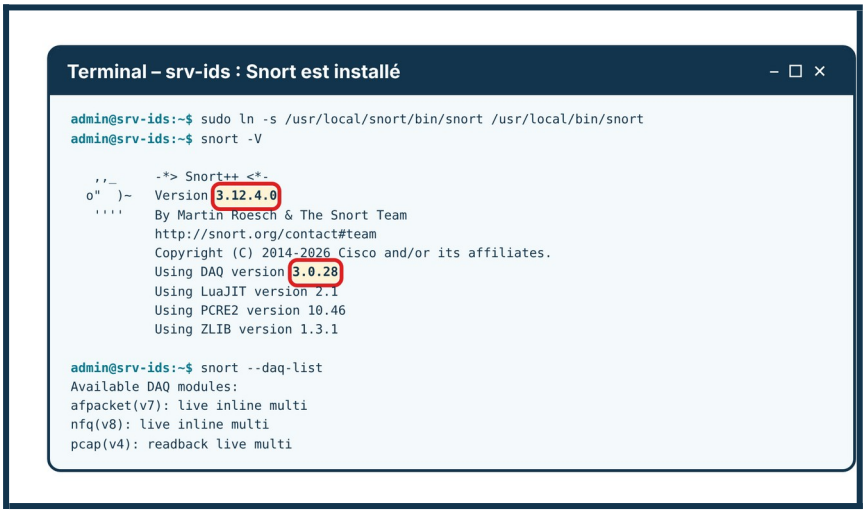
Figure 4 – Compilation et installation

D – Vérifier l'installation

1 Créez un raccourci pour appeler simplement `snort`, puis affichez la version et la liste des modules de capture :

TERMINAL

```
sudo ln -s /usr/local/snort/bin/snort /usr/local/bin/snort
snort -V
snort --daq-list
```



À SAVOIR

Si `snort --daq-list` répond « No available DAQ modules », Snort ne trouve pas les modules : vérifiez `ls /usr/local/lib/daq` et indiquez le dossier avec `--daq-dir /usr/local/lib/daq`. Si `snort` ne trouve pas `libdaq.so.3`, lancez `sudo ldconfig`.

Figure 5 – Version de Snort et modules DAQ

✔ **POINT DE CONTRÔLE**

- `snort -V` affiche **Version 3.12.4.0**
- `Using DAQ version 3.0.28` apparaît dans la bannière
- `snort --daq-list` liste au moins **afpacket** et **pcap**
- Vous savez expliquer pourquoi Snort est compilé plutôt qu'installé par `apt`

04 ÉTAPE 4 SUR 6 Configurer Snort et écrire ses premières règles

Le fichier `snort.lua` installé est un **modèle complet et commenté**. Nous n'y changeons que **trois choses** : le réseau protégé, la déclaration des règles et le journal des alertes. Faites d'abord une **copie de sauvegarde**, pour pouvoir revenir en arrière :

TERMINAL
```
sudo cp /usr/local/snort/etc/snort/snort.lua /usr/local/snort/etc/snort/snort.lua.orig
```

A – Déclarer le réseau protégé

1. Ouvrez le fichier avec `sudo nano /usr/local/snort/etc/snort/snort.lua`. Dans la section **1. configure defaults**, remplacez la valeur de `HOME_NET` par votre réseau (`EXTERNAL_NET` reste à `'any'`) :

snort.lua – section 1
```
HOME_NET = '192.168.20.0/24'
EXTERNAL_NET = 'any'
```

`HOME_NET` désigne **les machines à protéger**. Les règles s'y réfèrent par `$HOME_NET`, ce qui permet de les écrire une fois pour tous les réseaux. En laissant `EXTERNAL_NET` à `any`, Snort surveille aussi les attaques **venant de l'intérieur** (une machine compromise du VLAN), ce qui est précieux : `!$HOME_NET` ferait ignorer par les règles tout le trafic interne.

B – Déclarer les règles et activer les alertes internes

1. Dans la section **5. configure detection**, complétez le bloc `ips` (les lignes `enable_builtin_rules` et `include` sont déjà présentes sous forme de commentaires) :

snort.lua – section 5
```
ips =
{
    enable_builtin_rules = true,
    include = '/usr/local/snort/etc/rules/local.rules',
    variables = default_variables
}
```

Réglage	Rôle
`enable_builtin_rules = true`	Active les **alertes internes** de Snort : anomalies des décodeurs et des inspecteurs, dont la **détection de balayage de ports** (`port_scan`, déjà activée plus haut dans le fichier)
`include = '…/local.rules'`	Le **fichier de règles** à charger. Un fichier de règles peut lui-même en inclure d'autres avec une ligne `include` : nous nous en servirons pour les règles communautaires
`variables = default_variables`	Définit les variables utilisables dans les règles (`$HOME_NET`, `$HTTP_PORTS`…)

C – Choisir où écrire les alertes

1. Dans la section **7. configure outputs**, **décommentez** la ligne `alert_fast` et demandez l'écriture dans un fichier (sans cette option, les alertes s'affichent dans le terminal) :

```
snort.lua – section 7
alert_fast = { file = true }
```

Le format **alert_fast** est le plus lisible : **une ligne par alerte**. Le fichier est créé dans le dossier indiqué par l'option `-l` au lancement de Snort (ici `/var/log/snort`) et s'appelle `alert_fast.txt`.

D – Écrire les règles locales

1. Créez les dossiers puis le fichier de règles avec `sudo nano /usr/local/snort/etc/rules/local.rules` :

```
TERMINAL
sudo mkdir -p /usr/local/snort/etc/rules /var/log/snort
```

```
/usr/local/snort/etc/rules/local.rules
# Règles locales SIO – SRV-IDS
alert icmp any any -> $HOME_NET any (msg:"SIO ICMP echo"; itype:8; sid:1000001; rev:1;)
alert tcp any any -> $HOME_NET 22 (msg:"SIO SSH SYN"; flags:S; sid:1000002; rev:1;)
```

Une règle se lit de gauche à droite : un **en-tête** (qui parle à qui) puis des **options** entre parenthèses (quoi chercher). Décomposition de la première :

Élément	Valeur	Signification
Action	`alert`	Écrire une alerte (en IPS, `drop` ou `reject` bloquent le paquet)
Protocole	`icmp` / `tcp`	Famille de paquets concernée
Source	`any any`	N'importe quelle adresse, n'importe quel port
Direction	`->`	Du côté source vers le côté destination (la règle ne regarde pas le sens inverse)
Destination	`$HOME_NET any` / `$HOME_NET 22`	Une machine du réseau protégé (n'importe quel port, ou le port 22)
`msg`	`"SIO ICMP echo"`	Le texte affiché dans l'alerte
`itype:8`	type ICMP 8	Message ICMP « echo request » : c'est le **ping**
`flags:S`	drapeau TCP SYN	Premier paquet d'une connexion : une **tentative de connexion**
`sid` **et** `rev`	`1000001`, `1`	Identifiant et révision de la règle. **Les sid à partir de 1 000 000 sont réservés aux règles locales**



Figure 6 – Réglages de snort.lua et règles locales

À SAVOIR

`grep` n'affiche que les lignes utiles de `snort.lua`. Contrôlez que **chaque ligne modifiée n'est plus précédée de** `--` : en Lua, `--` introduit un commentaire.

E – Valider la configuration

1 Le mode test (`-T`) charge toute la configuration **sans écouter le réseau** : c'est le réflexe à avoir après chaque modification, **avant** de redémarrer le service :

TERMINAL

```
sudo snort -c /usr/local/snort/etc/snort/snort.lua -T
```

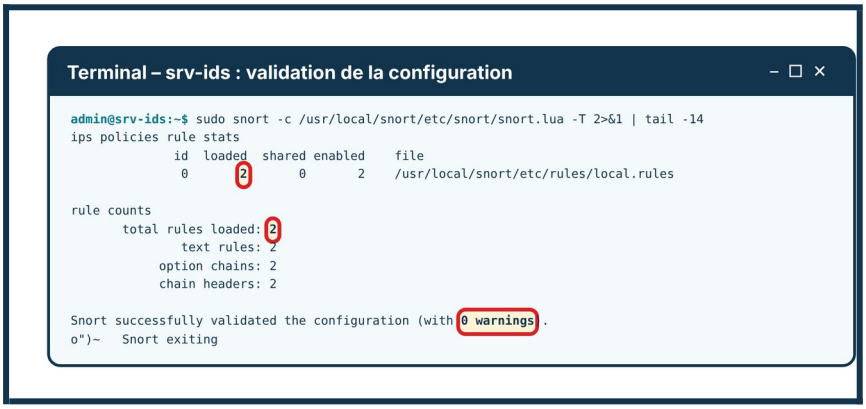


Figure 7 – Validation de la configuration

À SAVOIR

Le tableau « ips policies rule stats » indique combien de règles ont été lues dans chaque fichier : 2 pour nos deux règles locales. En cas d'erreur de syntaxe, Snort s'arrête en indiquant le **fichier et la ligne** fautive.

✔ **POINT DE CONTRÔLE**

- `HOME_NET` vaut **192.168.20.0/24** dans `snort.lua`
- Le fichier `local.rules` contient **deux règles** avec des **sid** uniques
- `snort -T` affiche « **successfully validated** » avec **0 warnings**
- Vous savez expliquer les parties d'une règle : action, en-tête, options

ÉTAPE 5 SUR 6

05 Détecter : ping, scan de ports et règles communautaires

A – Lancer Snort en écoute

Pour les tests, Snort tourne **au premier plan** dans un terminal. Sur **SRV-IDS**, lancez :

TERMINAL 1 (SRV-IDS)

```
sudo snort -c /usr/local/snort/etc/snort/snort.lua --daq afpacket -i ens18 -k none -l /var/log/snort -q
```

Option	Rôle
`-c …/snort.lua`	Le fichier de configuration à charger
`--daq afpacket`	Module de capture : lecture directe de la carte (par défaut Snort utiliserait `pcap`)
`-i ens18`	L'interface réseau à écouter
`-k none`	N'**ignore pas** les paquets dont la somme de contrôle semble fausse : la carte calcule les sommes elle-même, et les paquets émis par la machine locale en paraissent à tort invalides
`-l /var/log/snort`	Le dossier des journaux, où sera écrit `alert_fast.txt`
`-q`	Mode silencieux : pas de statistiques à l'écran

Le terminal reste occupé : Snort écoute (arrêtez-le avec **Ctrl+C**). Ouvrez une **seconde session** sur SRV-IDS et suivez le fichier d'alertes (`-F` attend qu'il soit créé) :

TERMINAL 2 (SRV-IDS)

```
sudo tail -F /var/log/snort/alert_fast.txt
```

B – Générer du trafic de test

Depuis **srv-lnx-01**, envoyez un ping, puis un **scan de ports** avec nmap vers SRV-IDS (installez-le avec `sudo apt install -y nmap`). L'option `-sS` envoie des paquets SYN sans terminer les connexions :

TERMINAL (srv-lnx-01)

```
ping -c 3 192.168.20.69
sudo nmap -sS -p 1-1000 192.168.20.69
```

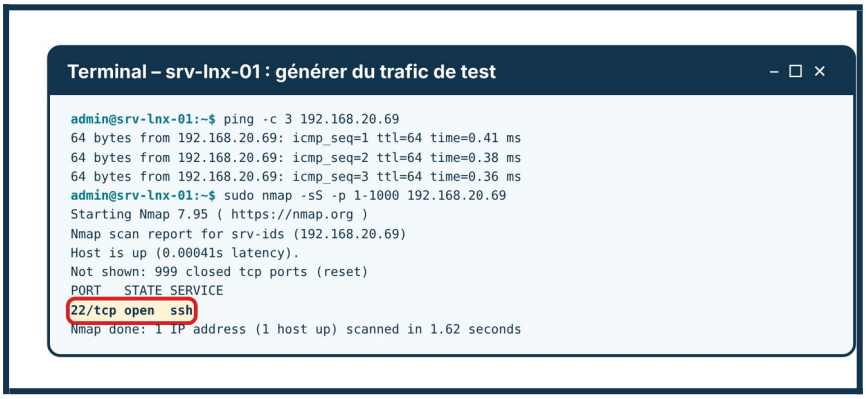


Figure 8 – Trafic de test émis par srv-lnx-01

À SAVOIR

Le scan parcourt les mille premiers ports de SRV-IDS : un seul est ouvert (SSH), les autres répondent « reset ». C'est exactement ce que fait un attaquant qui **cartographie** les services d'une machine avant de l'attaquer.

C – Lire les alertes

Dans le terminal 2, les alertes apparaissent aussitôt :

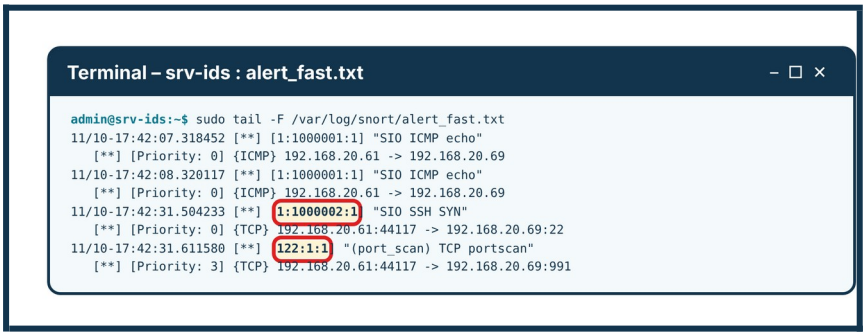


Figure 9 – Alertes écrites par Snort

À SAVOIR

Dans le fichier, chaque alerte tient sur **une seule ligne** ; elles sont coupées ici pour la lisibilité. Les trois ping déclenchent trois alertes ICMP, le SYN vers le port 22 la règle SSH, et le balayage la détection interne `port_scan`.

Champ	Exemple	Signification
Date et heure	`11/10-17:42:31.504233`	Mois/jour-heure, avec les microsecondes
`[gid:sid:rev]`	`[1:1000002:1]`	**gid** : la famille (1 = règle texte, **122** = détection de balayage de ports) ; **sid** : la règle ; **rev** : sa révision
Message	`"SIO SSH SYN"`	Le `msg` de la règle ; `(port_scan)` préfixe les alertes du module de balayage
`[Priority: n]`	`[Priority: 3]`	Gravité (1 = élevée) ; elle dépend du `classtype` de la règle
Protocole et flux	`{TCP} 192.168.20.61:44117 -> 192.168.20.69:22`	Protocole, source (adresse et port), destination

À SAVOIR

Aucune alerte ? Vérifiez, dans l'ordre : Snort tourne et écoute la bonne carte (`-i`) ; la machine de test envoie bien le trafic **vers l'adresse de SRV-IDS** ; `HOME_NET` contient l'adresse de destination ; `snort -T` annonce bien **2 règles chargées** ; les fonctions GRO/LRO sont désactivées ; et le sens de la règle (`->`) correspond au sens du trafic.

D – Ajouter les règles communautaires

Deux règles écrites à la main protègent peu. Le projet Snort publie gratuitement, **sans compte**, un jeu de **règles communautaires** qui couvre des milliers de comportements connus (exploits, scans, communications de logiciels malveillants…). Arrêtez Snort (Ctrl+C dans le terminal 1), puis téléchargez les règles et déclarez-les depuis `local.rules` :

TERMINAL (SRV-IDS)

```
cd /tmp
curl -fsSLO https://www.snort.org/downloads/community/snort3-community-rules.tar.gz
tar -xzf snort3-community-rules.tar.gz
sudo cp snort3-community-rules/snort3-community.rules /usr/local/snort/etc/rules/
echo 'include /usr/local/snort/etc/rules/snort3-community.rules' | sudo tee -a
/usr/local/snort/etc/rules/local.rules
sudo snort -c /usr/local/snort/etc/snort/snort.lua -T
```



Figure 10 – Règles communautaires chargées

À SAVOIR

Relancez ensuite Snort (étape A). Ces règles s'appliquent surtout au trafic **applicatif** (HTTP, DNS, SMB…) : elles ne réagissent pas aux ping de nos tests, mais à du trafic réel d'attaque ou rejoué. Quelques **avertissements** à la validation peuvent apparaître (règles pour des fonctions non activées) : sans erreur, ils sont sans conséquence.

E – Analyser une capture enregistrée

Snort sait aussi **relire un fichier de capture** (`.pcap`) : très utile pour tester les règles sur du trafic enregistré, ou pour analyser a posteriori une capture faite avec **tcpdump**, **Wireshark** ou **Sniffnet**. Aucun module DAQ à indiquer (installez d'abord tcpdump avec `sudo apt install -y tcpdump`) :

TERMINAL (SRV-IDS)

```
sudo tcpdump -i ens18 -c 200 -w /tmp/test.pcap
mkdir -p /tmp/snort-out
sudo snort -c /usr/local/snort/etc/snort/snort.lua -r /tmp/test.pcap -l /tmp/snort-out -q
cat /tmp/snort-out/alert_fast.txt
```

✔ **POINT DE CONTRÔLE**

- Un **ping** depuis srv-lnx-01 produit des alertes **SIO ICMP echo**
- Le **scan nmap** produit l'alerte **(port_scan) TCP portscan** (gid **122**)
- Vous savez lire une ligne **alert_fast** : date, gid:sid:rev, message, priorité, flux
- La configuration avec les règles communautaires est **validée** par `snort -T`

06

ÉTAPE 6 SUR 6

Exploiter : service, journaux et mises à jour

A – Faire tourner Snort comme un service

Une sonde doit démarrer **toute seule** avec la machine, et ne pas tourner en `root` : si Snort était compromis par un paquet piégé, l'attaquant hériterait de ses droits. Nous créons un **utilisateur dédié** sans connexion possible, propriétaire du dossier des journaux, et nous ne lui donnons que les deux **capacités** nécessaires à la capture réseau :

TERMINAL

```
sudo useradd --system --no-create-home --shell /usr/sbin/nologin snort
sudo chown -R snort:snort /var/log/snort
```

1 Créez le fichier du service avec `sudo nano /etc/systemd/system/snort.service` :

/etc/systemd/system/snort.service

```
[Unit]
Description=Snort 3 — sonde de détection d'intrusion
After=network-online.target
Wants=network-online.target

[Service]
Type=simple
User=snort
Group=snort
AmbientCapabilities=CAP_NET_RAW CAP_NET_ADMIN
CapabilityBoundingSet=CAP_NET_RAW CAP_NET_ADMIN
ExecStartPre=/usr/local/snort/bin/snort -c /usr/local/snort/etc/snort/snort.lua -T -q
ExecStart=/usr/local/snort/bin/snort -c /usr/local/snort/etc/snort/snort.lua \
  --daq afpacket -i ens18 -k none -l /var/log/snort -q
Restart=on-failure
RestartSec=5

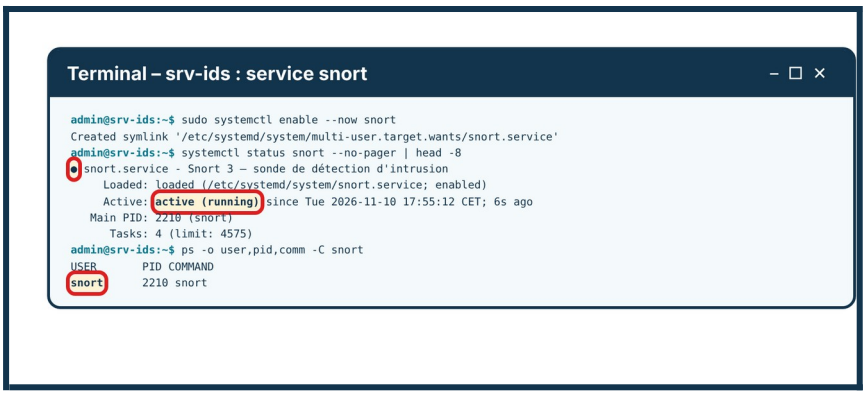
[Install]
WantedBy=multi-user.target
```

Ligne	Rôle
`User=snort`, `Group=snort`	Le service tourne avec l'utilisateur dédié, **pas en root**
`AmbientCapabilities=CAP_NET_RAW CAP_NET_ADMIN`	Donne **seulement** le droit d'ouvrir la carte en capture brute et de la passer en mode « promiscuous »
`ExecStartPre=… -T -q`	**Teste la configuration avant chaque démarrage** : une règle erronée empêche le démarrage au lieu de laisser une sonde muette
`Restart=on-failure`	Relance Snort après un plantage, 5 secondes plus tard

2 Activez et démarrez le service, puis contrôlez son état et l'utilisateur qui l'exécute :

TERMINAL

```
sudo systemctl daemon-reload
sudo systemctl enable --now snort
systemctl status snort --no-pager
ps -o user,pid,comm -C snort
```



À SAVOIR

En cas d'échec, lisez le journal du service : `journalctl -u snort -n 30 --no-pager`. Si Snort refuse de capturer sans être `root`, retirez les lignes `User=`, `Group=` et `AmbientCapabilities=`, et ajoutez `-u snort -g snort` à `ExecStart` : Snort démarre en `root`, ouvre la carte, puis **abandonne ses droits** pour l'utilisateur `snort`.

Figure 11 – Service snort actif

B – Faire tourner les journaux

Le fichier `alert_fast.txt` grossit sans fin : une sonde sur un réseau actif écrit **des milliers de lignes par jour**. `logrotate` l'archive chaque semaine et garde huit semaines. L'option `copytruncate` permet de le faire **sans arrêter Snort** :

/etc/logrotate.d/snort

```
/var/log/snort/alert_fast.txt {
    weekly
    rotate 8
    compress
    missingok
    notifempty
    copytruncate
}
```

1 Testez la configuration sans rien modifier (`-d` simule) : `sudo logrotate -d /etc/logrotate.d/snort`

C – Mettre à jour les règles

Les règles évoluent chaque semaine : de nouvelles menaces apparaissent, d'anciennes règles génèrent trop de faux positifs. Le script suivant **télécharge** les règles communautaires, **valide** la configuration avec les nouvelles règles et ne redémarre Snort que si tout est correct ; sinon il **remet l'ancien fichier** :

/usr/local/sbin/maj-regles-snort.sh

```
#!/bin/bash
# Met à jour les règles communautaires de Snort 3, valide puis redémarre le service
set -euo pipefail
RULES=/usr/local/snort/etc/rules
TMP=$(mktemp -d)
trap 'rm -rf "$TMP"' EXIT
echo "Téléchargement des règles communautaires..."
curl -fsSL -o "$TMP/rules.tar.gz" https://www.snort.org/downloads/community/snort3-
community-rules.tar.gz
tar -xzf "$TMP/rules.tar.gz" -C "$TMP"
cp "$RULES/snort3-community.rules" "$TMP/ancien.rules"
cp "$TMP/snort3-community-rules/snort3-community.rules" "$RULES/snort3-community.rules"
echo "Validation de la configuration..."
if /usr/local/snort/bin/snort -c /usr/local/snort/etc/snort/snort.lua -T -q; then
    echo "Configuration valide : redémarrage de Snort"
    systemctl restart snort
else
    echo "Configuration invalide : retour aux règles précédentes" >&2
    cp "$TMP/ancien.rules" "$RULES/snort3-community.rules"
    exit 1
fi
```

1. Rendez le script exécutable, essayez-le une fois à la main, puis planifiez-le chaque lundi à 3 h 30 avec le fichier `/etc/cron.d/snort-regles` :

TERMINAL

```
sudo chmod 750 /usr/local/sbin/maj-regles-snort.sh
sudo /usr/local/sbin/maj-regles-snort.sh
echo '30 3 * * 1 root /usr/local/sbin/maj-regles-snort.sh >> /var/log/snort/maj.log 2>&1'
| sudo tee /etc/cron.d/snort-regles
```

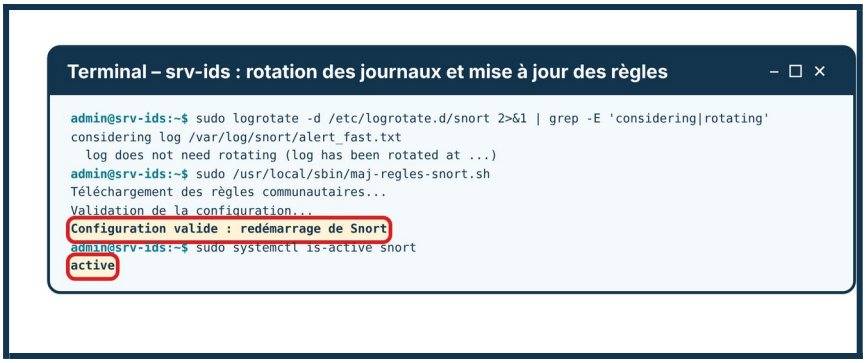


Figure 12 – Rotation des journaux et mise à jour des règles

À SAVOIR

Les règles communautaires sont un point de départ. Cisco Talos publie aussi un jeu **plus complet et plus réactif** (« Snort Subscriber Ruleset »), accessible avec un compte sur snort.org et un code d'accès personnel (**oinkcode**) ; l'outil **PulledPork 3** automatise leur téléchargement et la gestion des règles activées ou désactivées.

D – Réduire le bruit

Une sonde fraîchement installée **alerte beaucoup**, y compris pour des comportements parfaitement normaux. Il faut donc **régler** les règles : c'est le vrai travail d'exploitation d'un IDS. Trois leviers, dans la section **6. configure filters** de `snort.lua` ou dans les règles :

Levier	Exemple	Effet
`suppress`	`{ gid = 1, sid = 1000002, track = 'by_src', ip = '192.168.20.60' }`	Ignore une règle pour une adresse précise (par exemple un outil d'inventaire qui scanne légitimement)
`event_filter`	`{ gid = 1, sid = 1000001, type = 'limit', track = 'by_src', count = 1, seconds = 60 }`	Limite le nombre d'alertes d'une règle (ici une par minute et par source)
Désactiver une règle	Mettre `#` devant la ligne dans le fichier de règles	Retire la règle ; à documenter pour s'en souvenir

E – Passer à l'IPS : un aperçu

Pour **bloquer** et non plus seulement alerter, la sonde doit être **en coupure** : le trafic entre par une interface et ressort par une autre, avec Snort au milieu. On lance Snort en mode **inline** sur deux interfaces (`--daq afpacket -i ens19:ens20 -Q`), et les règles utilisent les actions `drop` ou `reject` à la place de `alert`. **N'activez le blocage qu'après des semaines d'IDS** : une seule règle trop large peut couper un service vital. Ce passage ne se fait pas sur le réseau de production sans plan de repli (liaison de contournement, ou « bypass », en cas de panne de la sonde).

F – Dépanner

Symptôme	Cause probable	Solution
`snort: command not found`	Raccourci absent	`sudo ln -s /usr/local/snort/bin/snort /usr/local/bin/snort` ou chemin complet
`libdaq.so.3: cannot open shared object file`	Cache des bibliothèques non actualisé	`sudo ldconfig`
`Could not find DAQ` au `configure_cmake.sh`	LibDAQ non installée ou introuvable	Reprendre l'étape 2 ; sinon `--with-daq-includes=/usr/local/include --with-daq-libraries=/usr/local/lib`
Compilation « Killed »	Mémoire insuffisante	`make -j1`, ajouter de la mémoire ou du swap
`snort -T` **signale une erreur de syntaxe**	Faute dans `snort.lua` ou une règle	Lire le fichier et la ligne indiqués ; revenir à `snort.lua.orig` si besoin
Snort tourne mais aucune alerte	Mauvaise interface, trafic non vu, `HOME_NET` erroné	Contrôler `-i`, le miroir de port, les règles chargées (`-T`), GRO/LRO
Alertes pour des paquets visiblement valides	Sommes de contrôle ou regroupement de la carte	`-k none` et `ethtool -K ens18 gro off lro off`
Le service ne démarre pas	Droits ou erreur de configuration	`journalctl -u snort` ; droits du dossier `/var/log/snort` ; variante `-u snort -g snort`
Perte de paquets, processeur saturé	Trafic trop important pour la machine	Plus de cœurs, **Hyperscan**, règles inutiles retirées

Pour aller plus loin

Snort 3 peut s'exécuter sur **plusieurs cœurs** (`-z`), journaliser en **JSON** pour un SIEM, détecter des **applications** (AppID) et analyser les **fichiers** transférés. Pour centraliser et visualiser les alertes, on les envoie vers une pile **Elastic**, **Wazuh** ou **Graylog** ; pour superviser la sonde elle-même, on la déclare dans **Zabbix** ou **Beszel**, et on vérifie sa disponibilité avec **Uptime Kuma**. Autres outils complémentaires : **Suricata** (autre IDS libre, avec des règles compatibles) et **Zeek** (analyse de protocoles).

✔ POINT DE CONTRÔLE

- Snort tourne comme **service** avec l'utilisateur **snort**, et démarre avec la machine
- Les journaux sont **archivés** par logrotate
- Les règles se mettent à jour avec le **script** et la configuration est validée avant redémarrage
- Vous savez expliquer la différence entre **IDS** et **IPS** et les risques du blocage
- Vous savez **dépanner** une sonde silencieuse (interface, miroir de port, GRO/LRO, règles)