

NOUVEAU

TUTORIEL PRATIQUE



INSTALLER ET CONFIGURER SNORT 3.11.1.0 SUR DEBIAN 13

De l'installation de base aux premières manipulations

POINTS CLES DU TUTORIEL



SNORT : QU'EST-CE QUE C'EST ?



INSTALLATION PAS A PAS DE SNORT (version 3.11.1.0)



TESTS D'ALERTE SNORT



POUR ALLER PLUS LOIN... CREATION D'UN DASHBOARD EN PHP

Notes :

Ce tutoriel décrit l'installation de SNORT sur une machine Debian 13.3. L'installation est présentée sous la forme d'une machine virtuelle créée à partir de l'hyperviseur © Proxmox.

Dans ce tutoriel, nous supposons que la machine virtuelle est connectée à l'interface « LAN » de votre pare-feu et qu'elle a obtenu une adresse IP dynamique et à accès à Internet.

1 – SNORT : QU'EST-CE QUE C'EST ?

Snort est un **système de détection d'intrusion réseau (NIDS — Network Intrusion Detection System)** open source, créé en 1998 par Martin Roesch. Racheté par Sourcefire en 2008, puis par Cisco en 2013, il reste aujourd'hui l'un des logiciels de sécurité réseau les plus répandus au monde.

Snort fonctionne en analysant le trafic réseau en temps réel et en le comparant à une base de signatures (règles). Lorsqu'un paquet ou une séquence de paquets correspond à une règle connue, Snort génère une **alerte**.

Les trois modes de fonctionnement de SNORT

Mode	Description	Usage typique
Sniffer	Affiche les paquets en temps réel dans le terminal.	Diagnostic réseau rapide
Packet Logger	Enregistre les paquets sur disque pour analyse ultérieure.	Audit
NIDS (détection)	Analyse le trafic selon des règles et génère des alertes.	Surveillance en production

Architecture et moteur de Snort 3

Snort 3 (version actuelle) introduit une **architecture multi-thread** permettant de **traiter plusieurs flux réseau simultanément**. Son moteur repose sur plusieurs composants clés :

- **DAQ (Data Acquisition Library)** : couche d'abstraction pour la capture réseau (AF_PACKET, pcap, nfq...)
- **Décodeur de paquets** : décodage des protocoles (Ethernet, IP, TCP, UDP, ICMP, HTTP, DNS...)
- **Preprocesseurs (inspecteurs)** : normalisation et réassemblage des flux (Frag3, Stream5, HTTP Inspect...)
- **Moteur de règles** : correspondance des paquets avec les signatures (pattern matching via Hyperscan)
- **Plugins de sortie** : écriture des alertes (unified2, syslog, JSON, console...)

Snort 2 vs Snort 3

Ce guide cible Snort 3 (≥ 3.0), qui est la version maintenue activement. Snort 2 est en fin de vie. La syntaxe des règles est globalement compatible, mais la configuration est très différente. Le fichier de configuration « **snort.lua** » remplace l'ancien fichier « snort.conf ».

Fonctionnement des règles Snort

Une règle Snort est une **ligne structurée composée d'un en-tête et d'options**.

Exemple de règle :

alert tcp any any -> \$HOME_NET 22 (msg:"SSH brute force attempt"; flags:S; threshold:type both, track by_src, count 5, seconds 60; sid:1000001; rev:1;)

Champ	Valeur exemple	Signification
Action	alert	Que faire quand la règle correspond (alert, drop, log, pass)
Protocole	tcp	Protocole concerné (tcp, udp, icmp, ip)
Source IP	any	Adresse IP source (any = toutes)
Source Port	any	Port source
Direction	->	Sens du trafic (-> ou <>)
Dest IP	\$HOME_NET 22	Variable réseau définie dans la config
msg:	"SSH brute force"	Message affiché dans l'alerte
sid:	1000001	Identifiant unique de la règle (>=1000000 pour les règles locales)
rev:	1	Révision de la règle

Place de Snort dans l'infrastructure

Dans notre environnement externalisé (SIOsecure), la VM Snort sera positionnée en écoute sur le segment LAN, en réception d'une copie du trafic via le port mirroring du vSwitch Proxmox. Elle opère en mode passif (IDS) : elle observe sans interférer avec le trafic.

Configuration de l'interface réseau en mode « promiscuité »

Pour que **Snort** puisse analyser l'ensemble du trafic réseau sur une interface, **il est souvent nécessaire de placer l'interface réseau en mode promiscuous (ou mode promisc)**. Cette configuration est essentielle pour un système de détection d'intrusion, car **elle permet de capturer tous les paquets qui passent par l'interface**, même ceux qui ne lui sont pas directement destinés. Voyons en détail ce qu'est le mode promisc et comment l'activer pour une interface réseau.

Qu'est-ce que le mode « PROMISC » ?

En fonctionnement normal, une carte réseau filtre le trafic pour ne traiter que les paquets adressés spécifiquement à son adresse MAC ou diffusés en **broadcast**. Autrement dit, elle ignore les paquets destinés aux autres adresses du réseau, ce qui est généralement suffisant pour des usages standards.

En revanche, en « **mode promiscuous** » (ou mode « promisc »), l'interface réseau capture **tous les paquets** qui transitent, peu importe leur destination. Cette configuration est idéale pour des outils de surveillance réseau comme **Snort**, car elle permet d'obtenir une vue complète du trafic circulant sur le réseau. Cela aide **Snort** à détecter des menaces, même si celles-ci visent d'autres hôtes que celui sur lequel Snort est installé.

A propos de « l'offloading »

Les cartes réseau modernes intègrent des fonctionnalités matérielles (« offloading ») pour alléger le travail du CPU. Parmi elles, **Generic Receive Offload (GRO)** et **Large Receive Offload (LRO)** sont problématiques pour Snort .

Ces fonctionnalités réassemblent les fragments de paquets en paquets plus gros avant de les transmettre au système d'exploitation. Pour un IDS/IPS comme Snort, cela pose deux problèmes majeurs :

- **Perte de visibilité** : Snort ne voit plus les petits fragments individuels, ce qui peut lui faire rater des attaques conçues pour se cacher dans la fragmentation.
- **Problèmes d'analyse** : Les gros paquets reconstitués peuvent dépasser la taille maximale analysable par Snort (snaplen) et perturber le moteur de reassemblage de flux (Stream), entraînant des erreurs d'analyse et des faux négatifs.

La recommandation officielle de l'équipe Snort est donc de désactiver GRO et LRO sur l'interface dédiée à l'inspection.

2 – INSTALLATION DE SNORT SUR DEBIAN 13

Etape 1 : mise à jour des paquets sur la machine Debian

apt update && apt full-upgrade -y

Etape 2 : installer « ethtool » et les dépendances nécessaires à Snort

« ethtool » est l'outil qui permet de configurer les paramètres de la carte réseau, y compris l'offloading.

Commande à copier-coller dans son intégralité (patientez pendant l'installation complète) :

```
sudo apt install -y ethtool build-essential autoconf automake libtool \  
libpcap-dev libpcr2-dev libdumbnet-dev bison flex zlib1g-dev \  
liblzma-dev openssl libssl-dev pkg-config libhwloc-dev \  
cmake liblua5.1-dev libunwind-dev libmnl-dev \  
libnetfilter-queue-dev libnhttp2-dev libhyperscan-dev \  
git wget unzip
```

Etape 3 : vérification de l'état actuel de « l'offloading »

Saisissez la commande ci-dessous en vérifiant, au préalable, que l'interface réseau de votre machine SNORT est référencée sous le nom « ens18 » sinon modifiez le nom de l'interface :

```
sudo ethtool -k ens18 | egrep 'large-receive-offload|generic-receive-offload|tcp-segmentation-offload'
```

Logiquement, le retour attendu devrait être comme ceci :

```
tcp-segmentation-offload: on  
generic-receive-offload: on  
large-receive-offload: off [fixed]
```

- **tcp-segmentation-offload (TSO)** : on → doit être désactivé.
- **generic-receive-offload (GRO)** : on → doit être désactivé.
- **Large-receive-offload (LRO)** : off [fixed] → déjà désactivé et ne peut pas être changé (c'est une caractéristique matérielle). Pas d'action nécessaire.

On désactive **TSO** et **GRO** avec la commande suivante :

```
sudo ethtool -K ens18 gro off tso off
```

On vérifie que les changements ont bien été pris en compte avec la commande suivante :

```
sudo ethtool -k ens18 | egrep 'large-receive-offload|generic-receive-offload|tcp-segmentation-offload'
```

Retour attendu :

```
tcp-segmentation-offload: off
generic-receive-offload: off
large-receive-offload: off [fixed]
```

On rend la désactivation permanente car si la machine redémarre, TSO et GRO seront réactivés ; pour cela on crée un service « ethtool-snort » avec la commande suivante :

```
sudo nano /etc/systemd/system/ethtool-snort.service
```

Coller le contenu suivant :

```
[Unit]
Description=Disable NIC offloading for Snort
Before=network.target snort.service
Wants=network.target

[Service]
Type=oneshot
ExecStart=/usr/sbin/ethtool -K ens18 gro off tso off
RemainAfterExit=yes

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

Activez le service avec les commandes suivantes :

```
sudo systemctl daemon-reload
sudo systemctl enable ethtool-snort.service
```

Etape 4 : configuration du mode « promiscuous »

Activez le mode « promisc » avec la commande suivante en vous assurant que l'interface réseau de votre machine SNORT est bien référencée sous le nom « ens18 » :

```
sudo ip link set ens18 promisc on
```

Editez le fichier de configuration réseau de votre machine Debian SNORT :

```
sudo nano /etc/network/interfaces
```

Ajoutez ces lignes à votre fichier :

```
up ip link set dev ens18 promisc on
down ip link set dev ens18 promisc off
```

Retour attendu :

```
# Activer le mode promiscuous au démarrage
up ip link set dev ens18 promisc on
down ip link set dev ens18 promisc off
```

Quittez et sauvegardez le fichier.

Faites redémarrer le service réseau avec la commande suivante :

```
sudo systemctl restart networking
```

Vérifiez que le mode promiscuité est bien actif sur l'interface de votre machine avec la commande suivante :

```
ip link show ens18 | grep PROMISC
```

Retour attendu :

```
prof@DEBIAN-SNORT-PROF:~$ ip link show ens18 | grep PROMISC
2: ens18: <BROADCAST,MULTICAST, PROMISC,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode DEFAULT group default qlen 1000
```

Etape 5 : compilation et installation des bibliothèques

1. Compilation et installation de SAFECLIB

Commencez par créer un répertoire de travail :

```
sudo mkdir -p ~/snort-build
cd ~/snort-build
```

Utilisez les commandes suivantes (dans votre répertoire « snort-build ») :

```
sudo wget https://github.com/rurban/safeclib/releases/download/v3.8.1/safeclib-3.8.1.tar.gz
```

```
sudo tar -xzf safeclib-3.8.1.tar.gz
```

```
cd safeclib-3.8.1.0-gdfea26
```

```
sudo ./configure --prefix=/usr/local
```

```
sudo make -j$(nproc)
```

```
sudo make install
```

Vérifiez l'installation avec la commande suivante :

```
ls -la /usr/local/lib | grep safec
```

Retour attendu :

```
prof@DEBIAN-SNORT-PROF:/safeclib-3.8.1.0-gdfea26$ ls -la /usr/local/lib | grep safec
-rw-r--r-- 1 root root 622492 1 mars 10:00 libsafec.a
-rwxr-xr-x 1 root root 945 1 mars 10:00 libsafec.la
lrwxrwxrwx 1 root root 17 1 mars 10:00 libsafec.so -> libsafec.so.3.0.8
lrwxrwxrwx 1 root root 17 1 mars 10:00 libsafec.so.3 -> libsafec.so.3.0.8
-rwxr-xr-x 1 root root 340176 1 mars 10:00 libsafec.so.3.0.8
```

Mettez à jour le cache des bibliothèques avec la commande suivante :

```
sudo ldconfig
```

2. Compilation et installation de DAQ

Depuis le répertoire « **snort-build** », saisissez les commandes suivantes :

```
sudo git clone https://github.com/snort3/libdaq.git && cd libdaq
sudo ./bootstrap
sudo ./configure --prefix=/usr/local
```

L'option `--prefix=/usr/local` indique que les fichiers seront installés dans `/usr/local` (binaires dans `/usr/local/bin`, bibliothèques dans `/usr/local/lib`, etc.). C'est l'emplacement standard pour les logiciels compilés manuellement.

Lancer la compilation avec la commande suivante :

```
sudo make -j$(nproc)
```

Installer avec la commande suivante :

```
sudo make install
```

Mettre à jour le cache des bibliothèques partagées avec la commande suivante :

```
sudo ldconfig
```

Vérifiez que le fichier de configuration « `pkg-config` » est bien présent avec la commande suivante :

```
ls -l /usr/local/lib/pkgconfig/libdaq.pc
```

Retour attendu :

```
prof@DEBIAN-SNORT-PROF:~/snort-build/safeclib-3.8.1.0-gdfea26/libdaq$ ls -l /usr/local/lib/pkgconfig/libdaq.pc
-rw-r--r-- 1 root root 291 1 mars 10:37 /usr/local/lib/pkgconfig/libdaq.pc
```

Si ce fichier n'existe pas, cela signifie que DAQ n'a pas généré le fichier `.pc`. Dans ce cas, nous devons le créer manuellement ou ajuster la variable `PKG_CONFIG_PATH`. Normalement, avec la version actuelle de DAQ, le fichier est bien installé. S'il est absent, vérifiez que le répertoire `/usr/local/lib/pkgconfig` existe et contient le fichier.

On définit la variable « `PKG_CONFIG_PATH` » pour que SNORT trouve DAQ. Même si le fichier est présent, il est possible que « `pkg-config` » ne le trouve pas par défaut. Nous allons donc exporter la variable d'environnement avec la commande suivante :

```
export PKG_CONFIG_PATH=/usr/local/lib/pkgconfig:$PKG_CONFIG_PATH
```

Pour que cette variable soit définie à chaque nouvelle session, vous pouvez l'ajouter à votre fichier « `~/bashrc` » avec les commandes suivantes :

```
echo 'export PKG_CONFIG_PATH=/usr/local/lib/pkgconfig:$PKG_CONFIG_PATH' >> ~/.bashrc
source ~/.bashrc
```

Maintenant que DAQ est installé, nous pouvons passer à l'étape suivante : la compilation et l'installation de Snort.

Etape 6 : compilation et installation de SNORT

Depuis le répertoire « **snort-build** », saisissez les commandes suivantes :

```
sudo wget https://github.com/snort3/snort3/archive/refs/tags/3.11.1.0.tar.gz
```

```
sudo tar -xzf 3.11.1.0.tar.gz && cd snort3-3.11.1.0
```

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

```
cd build
```

```
sudo make -j$(nproc)
```

Patiencez pendant la compilation qui va prendre du temps :

```
32%] Building CXX object src/service_inspectors/dnp3/CMakeFiles/dnp3.dir/dnp3.cc.o
32%] Built target dce_rpc
32%] Building CXX object src/service_inspectors/dnp3/CMakeFiles/dnp3.dir/dnp3_map.cc.o
32%] Building CXX object src/service_inspectors/dnp3/CMakeFiles/dnp3.dir/dnp3_module.cc.o
32%] Building CXX object src/service_inspectors/dnp3/CMakeFiles/dnp3.dir/dnp3_reassembly.cc.o
32%] Building CXX object src/service_inspectors/dnp3/CMakeFiles/dnp3.dir/dnp3_paf.cc.o
32%] Building CXX object src/service_inspectors/dnp3/CMakeFiles/dnp3.dir/ips_dnp3_data.cc.o
32%] Building CXX object src/service_inspectors/dnp3/CMakeFiles/dnp3.dir/ips_dnp3_func.cc.o
32%] Building CXX object src/service_inspectors/dnp3/CMakeFiles/dnp3.dir/ips_dnp3_ind.cc.o
32%] Building CXX object src/service_inspectors/dnp3/CMakeFiles/dnp3.dir/ips_dnp3_obj.cc.o
32%] Built target dnp3
34%] Building CXX object src/service_inspectors/dns/CMakeFiles/dns.dir/dns.cc.o
34%] Building CXX object src/service_inspectors/dns/CMakeFiles/dns.dir/dns_config.cc.o
34%] Building CXX object src/service_inspectors/dns/CMakeFiles/dns.dir/dns_module.cc.o
34%] Building CXX object src/service_inspectors/dns/CMakeFiles/dns.dir/dns_rr_decoder.cc.o
```

Lancez, ensuite, l'installation de SNORT avec la commande suivante :

```
sudo make install
```

```
sudo ldconfig
```

Vérifiez votre installation avec la commande suivante :

```
/usr/local/snort/bin/snort -V
```

Retour attendu :

```
prof@DEBIAN-SNORT-PROF:~/snort-build/snort3-3.11.1.0/build$ /usr/local/snort/bin/snort -V
o'-'~
o" )~
'-'~
'-'~
-*)> Snort++ <*-
Version 3.11.1.0
By Martin Roesch & The Snort Team
http://snort.org/contact#team
Copyright (C) 2014-2026 Cisco and/or its affiliates. All rights reserved.
Copyright (C) 1998-2013 Sourcefire, Inc., et al.
Using DAQ version 3.0.25
Using Hyperscan version 5.4.2 2024-12-30
Using libpcap version 1.10.5 (with TPACKET_V3)
Using LuaJIT version 2.1.1737090214
Using LZMA version 5.8.1
Using OpenSSL 3.5.4 30 Sep 2025
Using PCRE2 version 10.46 2025-08-27
Using ZLIB version 1.3.1
```

Maintenant que Snort est installé, nous pouvons passer à la configuration (création des répertoires, fichier snort.lua, règles, etc.).

Etape 1 : création des répertoires de base de SNORT

Snort a besoin de plusieurs répertoires pour stocker sa configuration, ses règles et ses logs. Nous allons les créer avec les permissions appropriées. Pour cela, utilisez les commandes suivantes :

```
sudo mkdir -p /etc/snort/rules
sudo mkdir -p /etc/snort/preproc_rules
sudo mkdir -p /var/log/snort
sudo mkdir -p /usr/local/lib/snort_dynamicrules
```

Explication :

- /etc/snort : contient les fichiers de configuration (snort.lua, etc.)
- /etc/snort/rules : contiendra vos règles personnalisées
- /etc/snort/preproc_rules : pour les règles des préprocesseurs (optionnel)
- /var/log/snort : emplacement des logs, alertes et fichiers PCAP
- /usr/local/lib/snort_dynamicrules : pour d'éventuelles règles dynamiques (non utilisé ici mais créé par défaut)

Etape 2 : création des fichiers de base

```
sudo touch /etc/snort/rules/local.rules
sudo touch /etc/snort/rules/classification.config
sudo touch /etc/snort/rules/reference.config
```

Explication :

- local.rules : vos règles personnalisées (nous y ajouterons des règles de test)
- classification.config : définit les types d'alertes et leurs priorités
- reference.config : pour référencer des sources externes (CVE, etc.)

Etape 3 : création du fichier de classification minimal

Explication : Ces lignes définissent deux classes d'alertes avec un niveau de priorité (1 = priorité haute, 2 = priorité moyenne). Snort s'attend à trouver ce fichier.

Copiez les commandes suivantes :

```
echo 'config classification: misc-attack,Misc Attack,1' | sudo tee /etc/snort/rules/classification.config
```

```
echo 'config classification: attempted-recon,Attempted Information Leak,2' | sudo tee -a /etc/snort/rules/classification.config
```

Etape 4 : création du fichier de configuration principal « snort.lua »

Snort, depuis la version 3, utilise un fichier au format « .lua » pour sa configuration. Nous allons créer un fichier de base, adapté à votre réseau.

```
sudo nano /etc/snort/snort.lua
```

Copiez le contenu du fichier donné en page suivante **en adaptant l'adresse de votre réseau local** !

Contenu à copier (**attention, ajustez « HOME_NET » à votre réseau local**) :

```
-- =====
-- snort.lua — Configuration Snort 3.11.1.0
-- Interface : ens18 | Réseau : XXX.XXX.XXX.0/24
-- =====

HOME_NET  = 'XXX.XXX.XXX.0/24'
EXTERNAL_NET = '!$HOME_NET'

-- Réassemblage des flux
stream    = {}
stream_tcp = { session_timeout = 180 }
stream_udp = { session_timeout = 180 }
stream_ip  = {}

-- Inspecteurs applicatifs
-- Avec Snort 3.11.1.0, la liaison port → service est gérée par wizard
-- Les inspecteurs ssh, telnet, dns ne prennent PAS de paramètres de port
wizard    = {}
http_inspect = {}
ftp_server = {}
ftp_client = {}
ftp_data   = {}
ssh        = {}
telnet     = {}
dns        = {}

-- Moteur IPS
ips =
{
    enable_builtin_rules = true,
    variables             = default_variables,
    include               = '/etc/snort/rules/local.rules',
}

-- Sortie des alertes
alert_fast =
{
    file = true,
    packet = false,
}
```

Quittez et sauvegardez le fichier.

Explication des points clés :

- **HOME_NET** : votre réseau local (remplacez si différent).
- **EXTERNAL_NET** = '!' .. **HOME_NET** : tout ce qui n'est pas le réseau local.
- **ips.include** : inclut vos règles personnalisées.

Etape 5 : ajout de règles de test

Pour vérifier que Snort fonctionne correctement, ajoutons quelques règles simples dans le fichier « **local.rules** ». Pour cela, éditez le fichier « local.rules » :

sudo nano /etc/snort/rules/local.rules

Collez le contenu suivant (**attention, adaptez votre adressage réseau !**) :

```
# =====
# local.rules — Snort 3.11.1.0
# Interface : ens18 | SSH sur port XXX
# =====

# --- ICMP ---
alert icmp any any -> XXX.XXX.XXX.0/24 any (msg:"ICMP Ping detecte"; itype:8; sid:1000001; rev:4;)
alert icmp any any -> XXX.XXX.XXX.0/24 any (msg:"ICMP Flood potentiel"; itype:8; detection_filter:track by_src, count 20, seconds 5; sid:1000002; rev:4;)

# --- SCANS DE PORTS ---
alert tcp any any -> XXX.XXX.XXX.0/24 any (msg:"SCAN SYN possible"; flags:S; flow:stateless; detection_filter:track by_src, count 20, seconds 3; sid:1000010; rev:4;)
alert tcp any any -> XXX.XXX.XXX.0/24 any (msg:"SCAN NULL"; flags:0; sid:1000011; rev:4;)
alert tcp any any -> XXX.XXX.XXX.0/24 any (msg:"SCAN XMAS"; flags:FPU; sid:1000012; rev:4;)
alert tcp any any -> XXX.XXX.XXX.0/24 any (msg:"SCAN FIN"; flags:F; sid:1000013; rev:4;)

# --- SSH (port personnalisé à adapter ici !) ---
alert tcp any any -> XXX.XXX.XXX.0/24 XXXXX (msg:"SSH Connexion entrante"; flow:to_server,established; content:"SSH-"; sid:1000020; rev:4;)
alert tcp any any -> XXX.XXX.XXX.0/24 XXXXX (msg:"SSH Brute force possible"; flow:to_server; detection_filter:track by_src, count 5, seconds 60; sid:1000021; rev:4;)

# --- HTTP ---
alert tcp any any -> XXX.XXX.XXX.0/24 80 (msg:"HTTP Traversee repertoire"; flow:to_server,established; content:"../"; http_uri; sid:1000030; rev:4;)
alert tcp any any -> XXX.XXX.XXX.0/24 80 (msg:"HTTP SQLi SELECT"; flow:to_server,established; content:"SELECT"; http_uri; sid:1000031; rev:4;)
alert tcp any any -> XXX.XXX.XXX.0/24 80 (msg:"HTTP SQLi UNION"; flow:to_server,established; content:"UNION"; http_uri; sid:1000032; rev:4;)
alert tcp any any -> XXX.XXX.XXX.0/24 80 (msg:"HTTP XSS script tag"; flow:to_server,established; content:"<script>"; http_uri; sid:1000033; rev:4;)

# --- DNS ---
alert udp any any -> XXX.XXX.XXX.0/24 53 (msg:"DNS Flood possible"; detection_filter:track by_src, count 50, seconds 10; sid:1000040; rev:4;)
alert udp XXX.XXX.XXX.0/24 53 -> any any (msg:"DNS Reponse volumineuse"; dsize:>512; sid:1000041; rev:4;)

# --- TELNET ---
alert tcp any any -> XXX.XXX.XXX.0/24 23 (msg:"TELNET Connexion - protocole non securise"; flow:to_server; sid:1000050; rev:4;)

# --- FTP ---
alert tcp any any -> XXX.XXX.XXX.0/24 21 (msg:"FTP Login anonyme"; flow:to_server,established; content:"USER anonymous"; sid:1000060; rev:4;)
alert tcp any any -> XXX.XXX.XXX.0/24 21 (msg:"FTP Tentative root"; flow:to_server,established; content:"USER root"; sid:1000061; rev:4;)
```

--- SMB / NetBIOS ---

alert tcp any any -> **XXX.XXX.XXX.0/24** 445 (msg:"SMB Connexion entrante"; flow:to_server; sid:1000080; rev:4;)

alert udp any any -> **XXX.XXX.XXX.0/24** 137 (msg:"NetBIOS Name Service"; sid:1000081; rev:4;)

Quittez et sauvegardez le fichier.

Etape 6 : attribution des permissions

Snort s'exécutera avec un **utilisateur « snort »** que nous allons créer pour des raisons de sécurité. Donnons-lui la propriété des répertoires de configuration et de logs avec les commandes suivantes :

```
sudo useradd -r -s /usr/sbin/nologin -d /var/log/snort snort
```

```
sudo chown -R snort:snort /etc/snort
```

```
sudo chown -R snort:snort /var/log/snort
```

```
sudo chmod 750 /etc/snort
```

```
sudo chmod 750 /var/log/snort
```

```
sudo chmod 640 /etc/snort/rules/*.rules
```

Ajoutez les capacités nécessaires avec la commande suivante :

```
sudo setcap cap_net_raw,cap_net_admin=eip /usr/local/snort/bin/snort
```

Vérifiez que les capacités sont bien ajoutées avec la commande suivante :

```
sudo getcap /usr/local/snort/bin/snort
```

Retour attendu :

```
/usr/local/snort/bin/snort cap_net_admin,cap_net_raw=eip
```

Etape 7 : tester la configuration

Tester votre configuration minimale avec la commande suivante ;

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

Retour attendu :

```
Finished /etc/snort/rules/local.rules:
-----
ips policies rule stats
      id loaded shared enabled file
      0  664      0      664 /etc/snort/snort.lua
-----
rule counts
  total rules loaded: 664
    text rules: 19
  builtin rules: 645
  option chains: 664
  chain headers: 11
-----
port rule counts
      tcp  udp  icmp  ip
any    649   0    2    0
src     0    1    0    0
dst     6    2    0    0
total  655   3    2    0
-----
service rule counts
      http:  to-srv  to-cli
      http2: 4      0
      http3: 4      0
      total: 12      0
-----
fast pattern groups
      dst: 4
      to_server: 3
-----
search engine (ac_bnfa)
  fast pattern only: 3
-----
pcap DAQ configured to passive.

Snort successfully validated the configuration (with 0 warnings).
o")~  Snort exiting
```

Etape 8 : Lancement en écoute sur l'interface « ens18 »

Ajoutez snort au « path » (recommandé) pour tous les utilisateurs avec les commandes suivantes :

```
echo 'export PATH=$PATH:/usr/local/snort/bin' | sudo tee /etc/profile.d/snort.env
source /etc/profile.d/snort.env
```

```
sudo ln -s /usr/local/snort/bin/snort /usr/local/bin/snort
```

Vérifiez que snort a bien été ajouté au path avec la commande suivante :

which snort

Retour attendu :

```
prof@DEBIAN-SNORT-PROF:/etc$ which snort
/usr/local/bin/snort
```

Saisissez la commande suivante pour lancer l'écoute sur l'interface réseau « ens18 » :

```
sudo snort -c /etc/snort/snort.lua -i ens18 -l /var/log/snort -D
```

Retour attendu :

```
Finished /etc/snort/rules/local.rules:
-----
ips policies rule stats
      id loaded shared enabled file
      0   664    0    664  /etc/snort/snort.lua
-----
rule counts
  total rules loaded: 664
    text rules: 19
  builtin rules: 664
  option chains: 664
  chain headers: 11
-----
port rule counts
      tcp      udp      icmp      ip
any    649      0      2      0
src      0      1      0      0
dst      6      2      0      0
total   655      3      2      0
-----
service rule counts
      http: 4 0
      http2: 4 0
      http3: 4 0
      total: 12 0
-----
fast pattern groups
      dst: 4
      to_server: 3
-----
search engine (ac_bnfa)
      instances: 5
      patterns: 15
      pattern chars: 98
      num states: 93
      num match states: 15
      memory scale: KB
      total memory: 8.5127
      pattern memory: 0.676758
      match list memory: 1.19531
      transition memory: 6.01562
      fast pattern only: 3
-----
pcap DAQ configured to passive.
initializing daemon mode
Commencing packet processing
child process is 42444
Retry queue interval is: 200 ms
++ [0] ens18
```

Etape 9 : tests

Pour suivre les alertes en temps réel, exécutez la commande suivante sur votre machine SNORT (CTRL+C pour stopper) :

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

Depuis une autre machine de votre réseau LAN, lancez un test de ping vers votre machine SNORT.

Résultat attendu (sur la machine SNORT) :

```
03/01-16:21:44.080005 [**] [1:1000001:4] "ICMP Ping detecte" [**] [Priority: 0] {ICMP} 10.10.10.106 -> 10.10.10.102
03/01-16:21:45.104107 [**] [1:1000001:4] "ICMP Ping detecte" [**] [Priority: 0] {ICMP} 10.10.10.106 -> 10.10.10.102
03/01-16:21:46.128107 [**] [1:1000001:4] "ICMP Ping detecte" [**] [Priority: 0] {ICMP} 10.10.10.106 -> 10.10.10.102
03/01-16:21:47.152203 [**] [1:1000001:4] "ICMP Ping detecte" [**] [Priority: 0] {ICMP} 10.10.10.106 -> 10.10.10.102
03/01-16:21:48.176100 [**] [1:1000001:4] "ICMP Ping detecte" [**] [Priority: 0] {ICMP} 10.10.10.106 -> 10.10.10.102
03/01-16:21:49.200189 [**] [1:1000001:4] "ICMP Ping detecte" [**] [Priority: 0] {ICMP} 10.10.10.106 -> 10.10.10.102
03/01-16:21:50.224136 [**] [1:1000001:4] "ICMP Ping detecte" [**] [Priority: 0] {ICMP} 10.10.10.106 -> 10.10.10.102
03/01-16:21:51.248303 [**] [1:1000001:4] "ICMP Ping detecte" [**] [Priority: 0] {ICMP} 10.10.10.106 -> 10.10.10.102
03/01-16:21:52.272451 [**] [1:1000001:4] "ICMP Ping detecte" [**] [Priority: 0] {ICMP} 10.10.10.106 -> 10.10.10.102
03/01-16:21:53.296599 [**] [1:1000001:4] "ICMP Ping detecte" [**] [Priority: 0] {ICMP} 10.10.10.106 -> 10.10.10.102
03/01-16:21:54.320747 [**] [1:1000001:4] "ICMP Ping detecte" [**] [Priority: 0] {ICMP} 10.10.10.106 -> 10.10.10.102
03/01-16:21:55.344895 [**] [1:1000001:4] "ICMP Ping detecte" [**] [Priority: 0] {ICMP} 10.10.10.106 -> 10.10.10.102
03/01-16:21:56.369043 [**] [1:1000001:4] "ICMP Ping detecte" [**] [Priority: 0] {ICMP} 10.10.10.106 -> 10.10.10.102
03/01-16:21:57.393191 [**] [1:1000001:4] "ICMP Ping detecte" [**] [Priority: 0] {ICMP} 10.10.10.106 -> 10.10.10.102
03/01-16:21:58.417339 [**] [1:1000001:4] "ICMP Ping detecte" [**] [Priority: 0] {ICMP} 10.10.10.106 -> 10.10.10.102
03/01-16:21:59.441487 [**] [1:1000001:4] "ICMP Ping detecte" [**] [Priority: 0] {ICMP} 10.10.10.106 -> 10.10.10.102
03/01-16:22:00.465635 [**] [1:1000001:4] "ICMP Ping detecte" [**] [Priority: 0] {ICMP} 10.10.10.106 -> 10.10.10.102
03/01-16:22:01.489783 [**] [1:1000001:4] "ICMP Ping detecte" [**] [Priority: 0] {ICMP} 10.10.10.106 -> 10.10.10.102
03/01-16:22:02.513931 [**] [1:1000001:4] "ICMP Ping detecte" [**] [Priority: 0] {ICMP} 10.10.10.106 -> 10.10.10.102
03/01-16:22:03.538079 [**] [1:1000001:4] "ICMP Ping detecte" [**] [Priority: 0] {ICMP} 10.10.10.106 -> 10.10.10.102
03/01-16:22:04.562227 [**] [1:1000001:4] "ICMP Ping detecte" [**] [Priority: 0] {ICMP} 10.10.10.106 -> 10.10.10.102
03/01-16:22:05.586375 [**] [1:1000001:4] "ICMP Ping detecte" [**] [Priority: 0] {ICMP} 10.10.10.106 -> 10.10.10.102
03/01-16:22:06.610523 [**] [1:1000001:4] "ICMP Ping detecte" [**] [Priority: 0] {ICMP} 10.10.10.106 -> 10.10.10.102
03/01-16:22:07.634671 [**] [1:1000001:4] "ICMP Ping detecte" [**] [Priority: 0] {ICMP} 10.10.10.106 -> 10.10.10.102
03/01-16:22:08.658819 [**] [1:1000001:4] "ICMP Ping detecte" [**] [Priority: 0] {ICMP} 10.10.10.106 -> 10.10.10.102
03/01-16:22:09.682967 [**] [1:1000001:4] "ICMP Ping detecte" [**] [Priority: 0] {ICMP} 10.10.10.106 -> 10.10.10.102
03/01-16:22:10.707115 [**] [1:1000001:4] "ICMP Ping detecte" [**] [Priority: 0] {ICMP} 10.10.10.106 -> 10.10.10.102
03/01-16:22:11.731263 [**] [1:1000001:4] "ICMP Ping detecte" [**] [Priority: 0] {ICMP} 10.10.10.106 -> 10.10.10.102
03/01-16:22:12.755411 [**] [1:1000001:4] "ICMP Ping detecte" [**] [Priority: 0] {ICMP} 10.10.10.106 -> 10.10.10.102
03/01-16:22:13.779559 [**] [1:1000001:4] "ICMP Ping detecte" [**] [Priority: 0] {ICMP} 10.10.10.106 -> 10.10.10.102
03/01-16:22:14.803707 [**] [1:1000001:4] "ICMP Ping detecte" [**] [Priority: 0] {ICMP} 10.10.10.106 -> 10.10.10.102
03/01-16:22:15.827855 [**] [1:1000001:4] "ICMP Ping detecte" [**] [Priority: 0] {ICMP} 10.10.10.106 -> 10.10.10.102
```

Dans cette capture, nous voyons également que nous sommes logués en SSH sur notre machine SNORT :

```
03/01-16:22:22.948881 [**] [1:1000021:4] "SSH Brute force possible" [**] [Priority: 0] {TCP} 90.58 :58118 -> 10.10.10.102:10
03/01-16:22:23.972983 [**] [1:1000001:4] "ICMP Ping detecte" [**] [Priority: 0] {ICMP} 10.10.10.106 -> 10.10.10.102
03/01-16:22:24.997085 [**] [1:1000001:4] "ICMP Ping detecte" [**] [Priority: 0] {ICMP} 10.10.10.106 -> 10.10.10.102
03/01-16:22:25.021187 [**] [1:1000001:4] "ICMP Ping detecte" [**] [Priority: 0] {ICMP} 10.10.10.106 -> 10.10.10.102
03/01-16:22:26.045289 [**] [1:1000001:4] "ICMP Ping detecte" [**] [Priority: 0] {ICMP} 10.10.10.106 -> 10.10.10.102
03/01-16:22:27.069391 [**] [1:1000001:4] "ICMP Ping detecte" [**] [Priority: 0] {ICMP} 10.10.10.106 -> 10.10.10.102
03/01-16:22:28.093493 [**] [1:1000001:4] "ICMP Ping detecte" [**] [Priority: 0] {ICMP} 10.10.10.106 -> 10.10.10.102
03/01-16:22:29.117595 [**] [1:1000001:4] "ICMP Ping detecte" [**] [Priority: 0] {ICMP} 10.10.10.106 -> 10.10.10.102
03/01-16:22:30.141697 [**] [1:1000001:4] "ICMP Ping detecte" [**] [Priority: 0] {ICMP} 10.10.10.106 -> 10.10.10.102
03/01-16:22:31.165799 [**] [1:1000001:4] "ICMP Ping detecte" [**] [Priority: 0] {ICMP} 10.10.10.106 -> 10.10.10.102
03/01-16:22:32.189901 [**] [1:1000001:4] "ICMP Ping detecte" [**] [Priority: 0] {ICMP} 10.10.10.106 -> 10.10.10.102
03/01-16:22:33.214003 [**] [1:1000001:4] "ICMP Ping detecte" [**] [Priority: 0] {ICMP} 10.10.10.106 -> 10.10.10.102
03/01-16:22:34.238105 [**] [1:1000001:4] "ICMP Ping detecte" [**] [Priority: 0] {ICMP} 10.10.10.106 -> 10.10.10.102
03/01-16:22:35.262207 [**] [1:1000001:4] "ICMP Ping detecte" [**] [Priority: 0] {ICMP} 10.10.10.106 -> 10.10.10.102
03/01-16:22:36.286309 [**] [1:1000001:4] "ICMP Ping detecte" [**] [Priority: 0] {ICMP} 10.10.10.106 -> 10.10.10.102
03/01-16:22:37.310411 [**] [1:1000001:4] "ICMP Ping detecte" [**] [Priority: 0] {ICMP} 10.10.10.106 -> 10.10.10.102
```

Commandes utiles à utiliser sur votre machine SNORT

Lancer l'écoute sur l'interface réseau (ici « ens18 »)

```
sudo snort -c /etc/snort/snort.lua -i ens18 -l /var/log/snort -D
```

Alertes en live (le plus utile)

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

20 dernières alertes

```
sudo tail -20 /var/log/snort/alert_fast.txt
```

Filtrer par type (ex: SSH)

```
sudo tail -f /var/log/snort/alert_fast.txt | grep SSH
```

Vérifier que le processus tourne

```
ps aux | grep snort
```

Tuer tous les processus Snort proprement

```
sudo kill -9 snort
```

```
sleep 2
```

Explications sur l'arrêt des processus (commande « pkill ») :

pkill -9

pkill envoie un **signal** à un processus. Le nombre après - est le numéro du signal :

Signal	Numéro	Nom	Comportement
pkill snort	15	SIGTERM	Demande « poliment » au processus de s'arrêter en fermant les fichiers ouverts et en vidant les buffers.
pkill -2 snort	2	SIGINT	Équivalent d'un Ctrl+C
pkill -9 snort	9	SIGKILL	Tue immédiatement le processus sans lui laisser le temps de rien faire : le noyau Linux force l'arrêt immédiat du processus.

En pratique pour Snort on utilise « -9 » car Snort peut ignorer SIGTERM quand il est occupé à traiter des paquets.

sleep 2

sleep 2 met en **pause le terminal pendant 2 secondes** avant d'exécuter la commande suivante.

On l'utilise ici parce qu'après pkill -9, le noyau Linux a besoin d'un court instant pour :

- libérer la mémoire occupée par Snort
- fermer les fichiers ouverts
- libérer l'interface réseau ens18

Sans ce délai, si on relance Snort immédiatement, il peut trouver l'interface encore "occupée" et refuser de démarrer.

Séquence complète expliquée :

sudo pkill -9 snort # tue immédiatement tous les processus snort

sleep 2 # attend 2 secondes que le noyau libère les ressources

ps aux | grep snort # permet d'afficher et de vérifier qu'il ne reste plus aucun processus snort actif sur la machine

3 – POUR ALLER PLUS LOIN... CREATION D'UN DASHBOARD

Etape 1 : modification du fichier « snort.lua » pour intégrer les alertes au format « json »

Dans cette partie, nous allons générer un script PHP qui lit les logs Snort et génère un tableau de bord des alertes SNORT. Pour cela, il faut activer la sortie « **json** » dans SNORT.

Éditez le fichier « **snort.lua** » avec la commande suivante :

sudo nano /etc/snort/snort.lua

Ajoutez le bloc suivant dans « -- **Sortie des alertes** » :

```
alert_json =
{
  file = true,
  limit = 100,
  fields = 'timestamp action msg proto src_addr src_port dst_addr dst_port sid rev',
}
```

Résultat attendu :

```
-- Sortie des alertes
alert_fast =
{
    file    = true,
    packet  = false,
}

alert_json =
{
    file    = true,
    limit   = 100,
    fields  = 'timestamp action msg proto src_addr src_port dst_addr dst_port sid rev',
}
```

Quittez et sauvegardez le fichier.

Appliquez les droits suivants au répertoire :

```
sudo chown snort:snort /var/log/snort
sudo chmod 750 /var/log/snort
sudo chmod 640 /var/log/snort/alert_json.txt
sudo chmod 640 /var/log/snort/alert_fast.txt
```

Tuez tous les processus SNORT en cours si besoin avec les commandes suivantes :

```
sudo pkill -9 snort
sleep 2
```

Lancez SNORT avec la commande suivante :

```
sudo -u snort snort -c /etc/snort/snort.lua -i ens18 -l /var/log/snort -D
```

Depuis une autre machine de votre réseau local, faites un test de ping en direction de la machine SNORT et vérifiez qu'un fichier « **alert_json.txt** » a bien été créé avec les commandes suivantes :

```
sudo ls -la /var/log/snort/
sudo tail -5 /var/log/snort/alert_json.txt
```

Résultat attendu :

```
total 4288
drwxrwx--- 2 snort snort    4096  2 mars  07:51 .
drwxr-xr-x 9 root  root    4096  2 mars  07:43 ..
-rw-r----- 1 snort snort 3731478  2 mars  08:08 alert_fast.txt
-rw-r----- 1 snort snort 642207  2 mars  08:08 alert_json.txt
```

Etape 2 : installation du serveur web Apache

Installez Apache et ses dépendances avec les commandes suivantes :

```
sudo apt install -y apache2 php libapache2-mod-php php-json
sudo systemctl enable apache2
sudo systemctl start apache2
```

Etape 3 : création de l'interface web

Créez les dossiers et fichiers de la future interface web avec les commandes suivantes :

```
sudo mkdir -p /var/www/snort-dashboard
sudo nano /var/www/snort-dashboard/index.php
```

Collez ce contenu :

```
<?php
// =====
// Snort 3 Dashboard — Interface légère pour BTS SIO
// Compatible alert_json Snort 3.11.1.0
// =====

$logFile = '/var/log/snort/alert_json.txt';
$maxAlerts = 200;
$alerts = [];

if (file_exists($logFile)) {
    $lines = array_reverse(array_filter(
        explode("\n", file_get_contents($logFile))
    ));
    foreach (array_slice($lines, 0, $maxAlerts) as $line) {
        $data = json_decode(trim($line), true);
        if ($data) $alerts[] = $data;
    }
}

// Statistiques
$total = count($alerts);
$byMsg = array_count_values(array_column($alerts, 'msg'));
$bySrc = array_count_values(array_column($alerts, 'src_addr'));
$byProto = array_count_values(array_column($alerts, 'proto'));
arsort($byMsg); arsort($bySrc);
$topMsg = array_slice($byMsg, 0, 5, true);
$topSrc = array_slice($bySrc, 0, 5, true);
?>
<!DOCTYPE html>
<html lang="fr">
<head>
<meta charset="UTF-8">
<meta http-equiv="refresh" content="30">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>Snort 3 — Dashboard</title>
<style>
* { box-sizing: border-box; margin: 0; padding: 0; }
body { font-family: 'Segoe UI', Arial, sans-serif; background: #0f1117; color: #e0e0e0; }

header {
    background: linear-gradient(135deg, #1a1f35, #0d3b6e);
    padding: 20px 30px;
    display: flex; align-items: center; justify-content: space-between;
    border-bottom: 2px solid #2e75b6;
}
header h1 { font-size: 1.6em; color: #fff; letter-spacing: 1px; }
```

```

header h1 span { color: #4da6ff; }
.badge {
  background: #e74c3c; color: #fff;
  padding: 6px 16px; border-radius: 20px; font-size: 0.85em; font-weight: bold;
}
.refresh { color: #888; font-size: 0.8em; margin-top: 4px; }

.container { max-width: 1400px; margin: 0 auto; padding: 24px; }

.cards { display: grid; grid-template-columns: repeat(auto-fit, minmax(200px, 1fr)); gap: 16px; margin-bottom: 24px; }
.card {
  background: #1a1f2e; border-radius: 10px; padding: 20px;
  border-left: 4px solid #2e75b6; text-align: center;
}
.card.red { border-color: #e74c3c; }
.card.green { border-color: #2ecc71; }
.card.orange { border-color: #f39c12; }
.card .num { font-size: 2.2em; font-weight: bold; color: #4da6ff; }
.card.red .num { color: #e74c3c; }
.card.green .num { color: #2ecc71; }
.card.orange .num { color: #f39c12; }
.card .label { font-size: 0.85em; color: #888; margin-top: 6px; text-transform: uppercase; letter-spacing: 1px; }

.grid2 { display: grid; grid-template-columns: 1fr 1fr; gap: 16px; margin-bottom: 24px; }
@media (max-width: 900px) { .grid2 { grid-template-columns: 1fr; } }

.panel { background: #1a1f2e; border-radius: 10px; padding: 20px; }
.panel h2 { font-size: 1em; color: #4da6ff; margin-bottom: 16px; text-transform: uppercase; letter-spacing: 1px;
border-bottom: 1px solid #2a2f45; padding-bottom: 10px; }

.bar-row { display: flex; align-items: center; margin-bottom: 10px; font-size: 0.88em; }
.bar-label { width: 200px; white-space: nowrap; overflow: hidden; text-overflow: ellipsis; color: #ccc; }
.bar-wrap { flex: 1; background: #2a2f45; border-radius: 4px; height: 18px; margin: 0 10px; }
.bar-fill { height: 100%; border-radius: 4px; background: linear-gradient(90deg, #2e75b6, #4da6ff); min-width: 4px;
}
.bar-count { color: #f39c12; font-weight: bold; min-width: 30px; text-align: right; }

table { width: 100%; border-collapse: collapse; font-size: 0.85em; }
thead th {
  background: #12162a; color: #4da6ff; padding: 10px 12px;
  text-align: left; font-size: 0.8em; text-transform: uppercase; letter-spacing: 1px;
  position: sticky; top: 0;
}
tbody tr { border-bottom: 1px solid #1e2438; transition: background 0.15s; }
tbody tr:hover { background: #1e2a42; }
td { padding: 8px 12px; color: #ccc; }
td.msg { color: #fff; font-weight: 500; max-width: 280px; }
td.proto { text-align: center; }

.proto-badge {
  display: inline-block; padding: 2px 8px; border-radius: 4px;
  font-size: 0.8em; font-weight: bold; text-transform: uppercase;
}
.proto-tcp { background: #1a3a5c; color: #4da6ff; }
.proto-udp { background: #1a4a2e; color: #2ecc71; }

```

```

.proto-icmp { background: #4a2e1a; color: #f39c12; }

.table-wrap { overflow-x: auto; max-height: 450px; overflow-y: auto; }
.section-title { font-size: 1em; color: #4da6ff; margin-bottom: 14px; text-transform: uppercase; letter-spacing: 1px;
}

footer { text-align: center; padding: 20px; color: #444; font-size: 0.8em; border-top: 1px solid #1a1f2e; margin-top: 10px; }
</style>
</head>
<body>

<header>
  <div>
    <h1>&#128737; Snort <span>3</span> — Dashboard</h1>
    <div class="refresh">Actualisation automatique toutes les 30s</div>
  </div>
  <div>
    <div class="badge"><?= $total ?> alertes</div>
    <div style="color:#888;font-size:0.75em;margin-top:4px;text-align:right"><?= date('d/m/Y H:i:s') ?></div>
  </div>
</header>

<div class="container">

  <!-- Cartes résumé -->
  <div class="cards">
    <div class="card red">
      <div class="num"><?= $total ?></div>
      <div class="label">Total alertes</div>
    </div>
    <div class="card">
      <div class="num"><?= count($byMsg) ?></div>
      <div class="label">Types distincts</div>
    </div>
    <div class="card orange">
      <div class="num"><?= count($bySrc) ?></div>
      <div class="label">IP sources uniques</div>
    </div>
    <div class="card green">
      <div class="num"><?= $byProto['TCP'] ?? 0 ?></div>
      <div class="label">Alertes TCP</div>
    </div>
    <div class="card green">
      <div class="num"><?= $byProto['UDP'] ?? 0 ?></div>
      <div class="label">Alertes UDP</div>
    </div>
    <div class="card orange">
      <div class="num"><?= $byProto['ICMP'] ?? 0 ?></div>
      <div class="label">Alertes ICMP</div>
    </div>
  </div>

  <!-- Top alertes + Top sources -->
  <div class="grid2">
    <div class="panel">

```

```

<h2>&#128202; Top 5 — Types d'alertes</h2>
<?php
$maxVal = max(array_values($stopMsg) ?: [1]);
foreach ($stopMsg as $msg => $cnt): ?>
<div class="bar-row">
    <div class="bar-label" title="<?= htmlspecialchars($msg) ?>"><?= htmlspecialchars($msg) ?></div>
    <div class="bar-wrap">
        <div class="bar-fill" style="width:<?= round(($cnt/$maxVal)*100) ?>%"></div>
    </div>
    <div class="bar-count"><?= $cnt ?></div>
</div>
<?php endforeach; ?>
</div>

<div class="panel">
<h2>&#127760; Top 5 — IP sources</h2>
<?php
$maxVal = max(array_values($stopSrc) ?: [1]);
foreach ($stopSrc as $ip => $cnt): ?>
<div class="bar-row">
    <div class="bar-label"><?= htmlspecialchars($ip ?: 'N/A') ?></div>
    <div class="bar-wrap">
        <div class="bar-fill" style="width:<?= round(($cnt/$maxVal)*100) ?>%;background:linear-
gradient(90deg,#c0392b,#e74c3c)"></div>
    </div>
    <div class="bar-count"><?= $cnt ?></div>
</div>
<?php endforeach; ?>
</div>
</div>

<!-- Journal des alertes -->
<div class="panel">
<h2>&#128196; Journal des <?= $total ?> dernières alertes</h2>
<div class="table-wrap">
<table>
<thead>
<tr>
<th>Horodatage</th>
<th>Message</th>
<th>Proto</th>
<th>Source</th>
<th>Port src</th>
<th>Destination</th>
<th>Port dst</th>
<th>SID</th>
</tr>
</thead>
<tbody>
<?php foreach ($alerts as $a): ?>
<tr>
<td><?= htmlspecialchars($a['timestamp'] ?? '-') ?></td>
<td class="msg"><?= htmlspecialchars($a['msg'] ?? '-') ?></td>
<td class="proto">
<?php $p = strtolower($a['proto'] ?? ''); ?>
<span class="proto-badge proto-<?= $p ?>"><?= strtoupper($p) ?></span>

```

```

</td>
<td><? = htmlspecialchars($a['src_addr'] ?? '-') ?></td>
<td><? = htmlspecialchars($a['src_port'] ?? '-') ?></td>
<td><? = htmlspecialchars($a['dst_addr'] ?? '-') ?></td>
<td><? = htmlspecialchars($a['dst_port'] ?? '-') ?></td>
<td style="color:#888"><? = htmlspecialchars($a['sid'] ?? '-') ?></td>
</tr>
<?php endforeach; ?>
</tbody>
</table>
</div>
</div>

</div>

<footer>Snort 3.11.1.0 — BTS SIO SISR — Interface générée le <? = date('d/m/Y à H:i:s') ?></footer>
</body>
</html>

```

Etape 4 : ajout des permissions nécessaires

Apache/PHP doit pouvoir lire « **/var/log/snort/alert_json.txt** ». Pour cela, ajoutez « **www-data** » au groupe « snort » avec la commande suivante :

```
sudo usermod -aG snort www-data
```

Assurez-vous que les bons droits en lecture au groupe soient affectés sur les logs avec les commandes suivantes :

```

sudo chown snort:snort /var/log/snort/alert_fast.txt
sudo chmod 750 /var/log/snort
sudo chmod 640 /var/log/snort/alert_json.txt
sudo chmod 640 /var/log/snort/alert_fast.txt

```

Redémarrez Apache pour prendre en compte le groupe avec la commande suivante :

```
sudo systemctl restart apache2
```

Etape 5 : créer un virtualhost pour votre sous-domaine « **snort.equipeX.siosecur.fr** »

Créez le fichier de configuration « snort.conf » avec la commande suivante :

```
sudo nano /etc/apache2/sites-available/snort.conf
```

Copiez le contenu suivant (**en adaptant votre sous-domaine !**) :

```

<VirtualHost *:80>
    ServerName snort.equipeX.siosecur.fr
    DocumentRoot /var/www/snort-dashboard

    <Directory /var/www/snort-dashboard>
        Options -Indexes -FollowSymLinks
        AllowOverride None
        Require all granted
    </Directory>

    ErrorLog ${APACHE_LOG_DIR}/snort_error.log

```

```
CustomLog ${APACHE_LOG_DIR}/snort_access.log combined
</VirtualHost>
```

Quittez et sauvegardez le fichier.

Activez le site et rechargez Apache avec les commandes suivantes :

```
sudo a2ensite snort.conf
```

Désactivez le site par défaut d'Apache avec la commande suivante :

```
sudo a2dissite 000-default.conf
```

Relancez Apache avec la commande suivante :

```
sudo systemctl restart apache2
```

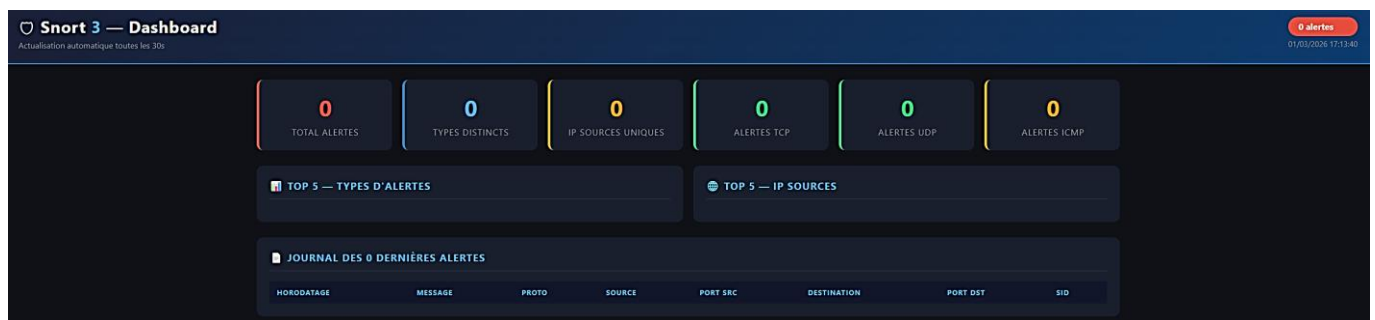
Etape 6 : création d'une règle de pare-feu

Dans votre pare-feu, créez une règle DNAT qui ouvre le port 80 à destination de votre machine SNORT.

Etape 7 : tests

Ouvrez un navigateur et saisissez votre URL : <https://snort.equipeX.siosecure.fr>

Vous devriez obtenir le tableau de bord suivant :



Rendez les droits permanents pour SNORT car à chaque redémarrage de la machine SNORT va recréer les fichiers d'alertes avec les droits « 600 ». Pour cela, ajoutez un « umask » dans le service « systemd » avec les commandes suivantes :

```
sudo nano /etc/systemd/system/snort.service
```

Copiez le contenu suivant :

[Unit]

Description=Snort 3 IDS

After=network.target ethtool-snort.service

[Service]

Type=forking

User=snort

Group=snort

UMask=0027

ExecStart=/usr/local/snort/bin/snort -c /etc/snort/snort.lua -i ens18 -l /var/log/snort -D

ExecStop=/bin/kill -SIGTERM \$MAINPID

Restart=on-failure

[Install]

WantedBy=multi-user.target

Quittez et sauvegardez le fichier.

Relancez le démon avec la commande suivante :

sudo systemctl daemon-reload

Note :

UMask=0027 force Snort à créer tous ses fichiers en 640 (snort:snort avec lecture pour le groupe) — [www-data](#) membre du groupe snort pourra donc toujours lire les logs, même après un redémarrage.

Avant de faire des tests de ping depuis une autre machine du réseau local, voici un rappel des commandes utiles pour l'utilisation rapide de SNORT :

Tuer tous les processus SNORT actifs

```
sudo kill -9 snort  
sleep 2
```

Vérifier les processus SNORT actifs

```
ps aux | grep snort
```

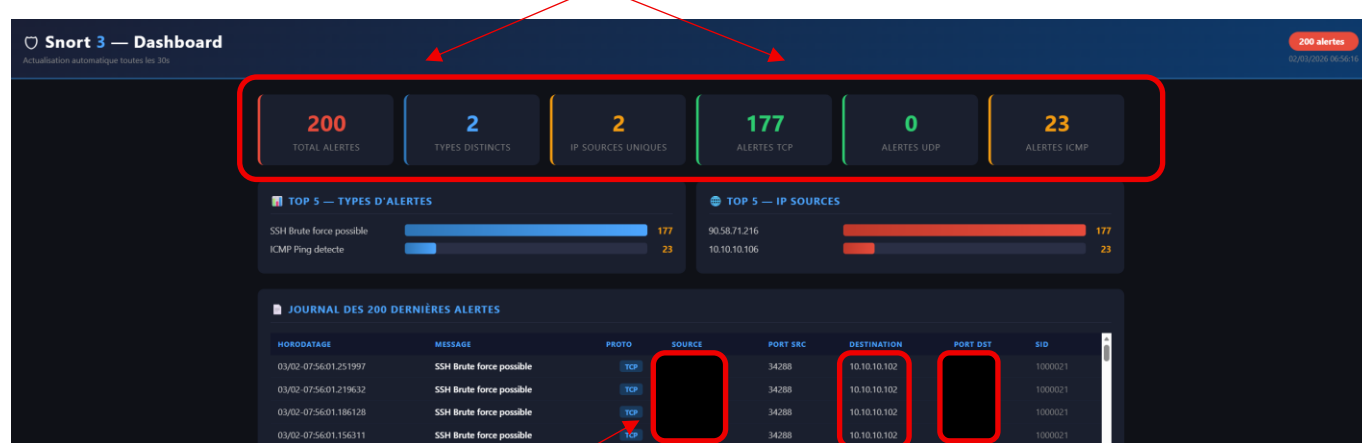
Vérifier que la configuration SNORT est valide

```
sudo -u snort snort -c /etc/snort/snort.lua -T
```

Relancer SNORT sur l'interface qu'il doit écouter (ens18)

```
sudo -u snort snort -c /etc/snort/snort.lua -i ens18 -l /var/log/snort -D
```

Depuis une machine du réseau local, lancez un test de ping vers la machine SNORT puis actualisez le Dashboard. Si tout est fonctionnel, vous devriez obtenir des alertes en temps réel comme ceci :



Ici s'affichera l'adresse IP source

ICI s'affichera le port SSH de la machine de destination