🛡️ Projet E6 – IDS Snort dans une infrastructure virtualisée

ⓘ Informations du projet

> **Étudiant :** Matthis FINOCCHI
>
> **Formation :** BTS SIO option SISR
>
> **Établissement :** UTEC Avon-Fontainebleau / Melun
>
> **Année :** 2025–2026

📋 Navigation du dossier technique

| # | Note | Description |

|---|---|---|

| 01 | 01 - Contexte du projet | Entreprise fictive & besoins sécurité |

| 02 | 02 - Objectifs du projet | Détection, surveillance, amélioration |

| 03 | 03 - Architecture technique | VMs, rôles, réseau |

| 04 | 04 - Tableau infrastructure | Récapitulatif machines |

| 05 | 05 - Schéma réseau | Topologie ASCII + draw.io |

| 06 | 06 - Installation Snort | Étapes d'installation complètes |

| 07 | 07 - Configuration Snort | snort.conf, interfaces, mode IDS |

| 08 | 08 - Règles Snort personnalisées | Règles ICMP, SSH, scan de ports... |

| 09 | 09 - Simulation d'attaques | Kali Linux – Nmap, Hydra, hping3 |

| 10 | 10 - Analyse des alertes | Lecture et interprétation des logs |

| 11 | 11 - Sécurisation et améliorations | Firewall, SIEM, Zabbix... |

| 12 | 12 - Compétences BTS SIO SISR | Référentiel de compétences |

| 13 | 13 - Conclusion | Bilan du projet |

🗺️ Mindmap du projet

```
Error parsing Mermaid diagram!

Parse error on line 3:
mindmaproot((IDS Snort))I
_______^
Expecting 'SPACELINE', 'NL', 'EOF', got 'NODE_ID'
```

📊 Avancement

- [x] ~~Contexte et objectifs rédigés~~
- [x] ~~Architecture définie~~
- [ ] Installation Snort réalisée
- [ ] Règles personnalisées créées
- [ ] Tests d'attaques effectués
- [ ] Rapport finalisé

Lié à : Projet Zabbix E6 | *Portfolio : mfiportfolio.fr*

---

tags:

- BTS-SIO
- SISR
- E6
- Snort
- Contexte

parents: "00 - Index du projet"

---

01 – Contexte du projet

🏢 Présentation de l'entreprise fictive

TechNova Solutions est une PME spécialisée dans le développement de logiciels de gestion pour le secteur industriel.

| Informations | Détails |

|---|---|

| **Secteur** | Édition de logiciels industriels |

| **Effectif** | 85 employés |

| **Sites** | Siège à Évry + antenne à Melun |

| **Fondée** | 2015 |

| **Chiffre d'affaires** | ~4,2 M€/an |

Infrastructure existante

- ~40 postes de travail Windows 10/11
- 3 serveurs physiques (fichiers, ERP, AD)
- Connexion Internet via un routeur pare-feu basique
- Pas de solution de supervision de sécurité en place
- Données sensibles : plans de fabrication, contrats clients, données financières

---

🔍 Besoins en sécurité réseau

Contexte déclencheur

Suite à une **tentative d'intrusion détectée fortuitement** lors d'une maintenance réseau, la direction informatique a mandaté un audit de sécurité. Cet audit a révélé plusieurs lacunes critiques :

- Aucune visibilité sur le trafic réseau interne
- Pas de journalisation centralisée des événements de sécurité
- Délai de détection estimé à plusieurs jours après une compromission

Besoins identifiés

⚠ **Risques actuels**

> Sans solution IDS, TechNova est aveugle face aux attaques internes et aux latéralisations d'un attaquant déjà présent sur le réseau.

1. **Détection temps réel** des tentatives d'intrusion et comportements réseau anormaux
2. **Surveillance du trafic** entrant/sortant sur le réseau d'entreprise
3. **Alertes automatiques** à destination de l'équipe IT (email, syslog)
4. **Journalisation** des événements pour investigation forensique
5. **Solution open-source** maîtrisable et évolutive, sans coût de licence

Problématique

? **Problématique du projet**

> **Comment mettre en place un système de détection d'intrusion (IDS) efficace et configurable dans une infrastructure réseau virtualisée, tout en minimisant les faux positifs et en garantissant la lisibilité des alertes pour l'équipe IT ?**

🔗 Navigation

← Retour : 00 - Index du projet

→ Suite : 02 - Objectifs du projet

---

tags:

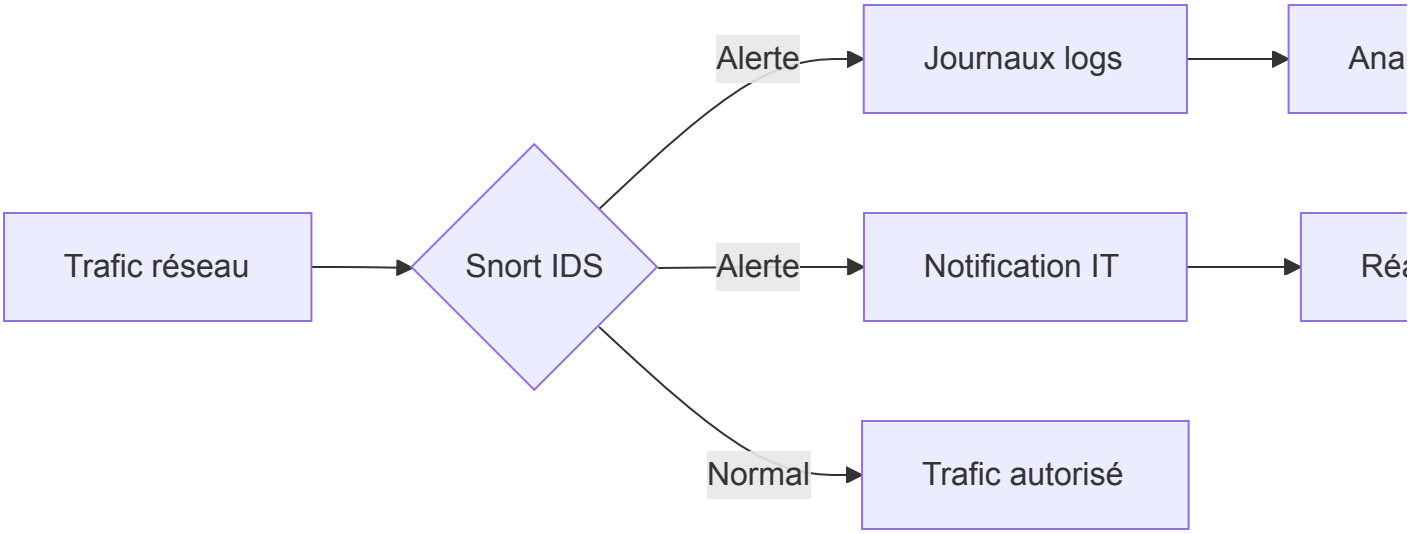
- BTS-SIO
- SISR
- E6
- Snort
- Objectifs

parents: "00 - Index du projet"

---

02 – Objectifs du projet

🎯 Vue d'ensemble



1️⃣ Détection d'intrusion

L'objectif principal est de déployer **Snort 3** en mode IDS (Intrusion Detection System) pour surveiller **passivement** le trafic réseau sans le bloquer.

Ce que Snort doit détecter

| Type d'attaque | Méthode | Priorité |

|---|---|---|

| Scan de ports | SYN scan Nmap | 🔴 Haute |

| Brute-force SSH | Hydra / Medusa | 🔴 Haute |

| Ping flood ICMP | hping3 | 🟡 Moyenne |

| Scan UDP | Nmap -sU | 🟡 Moyenne |

| Connexions FTP anonymes | Accès non authentifié | 🟡 Moyenne |

| Trafic SMB suspect | Enum4linux, smbclient | 🟢 Basse |

| Signatures malware connues | Règles Snort Community | 🔴 Haute |

2️⃣ Surveillance du trafic réseau

- Capturer et analyser le trafic en **temps réel** sur le segment 192.168.10.0/24
- Identifier les protocoles utilisés (TCP, UDP, ICMP, HTTP, SSH...)
- Distinguer trafic légitime vs trafic potentiellement malveillant
- Générer des statistiques de trafic observé

🔥 Mode promiscuous

> L'interface réseau du serveur Snort est configurée en mode **promiscuous** : elle capture *tous* les paquets du segment, même ceux qui ne lui sont pas adressés.

3️⃣ Amélioration de la sécurité

Indicateurs de succès (KPI)

| Indicateur | Objectif |

|---|---|

| MTTD (Mean Time To Detect) | < 5 minutes |

| Taux de faux positifs | < 10% |

| Couverture des attaques simulées | 100% détectées |

| Disponibilité du service Snort | > 99% |

Évolutivité prévue

- [ ] Snort IDS → Snort IPS (mode inline, blocage actif)
- [ ] Intégration avec un SIEM (Splunk, Wazuh)
- [ ] Supervision du service Snort via Zabbix
- [ ] Centralisation des logs (ELK Stack)

🔗 Navigation

← Retour : 01 - Contexte du projet

→ Suite : 03 - Architecture technique

---

tags:

- BTS-SIO
- SISR
- E6
- Snort
- Architecture
- Virtualisation

parents: "00 - Index du projet"

---

03 – Architecture technique

🖥️ Hyperviseur

L'infrastructure est entièrement virtualisée. Deux options selon l'environnement :

| Environnement | Hyperviseur | Réseau |

|---|---|---|

| **Maison (Mac)** | VMware Fusion / UTM | Host-only ou Internal |

| **École / Labo** | Proxmox VE | Bridge interne vmbr1 |

Le réseau utilisé est **192.168.10.0/24** — isolé, sans routage vers Internet depuis les VMs cibles.

📦 Machines virtuelles

snort-ids — Serveur IDS

| Propriété | Valeur |

|---|---|

| **OS** | Ubuntu Server 22.04 LTS |

| **IP** | 192.168.10.10/24 |

| **vCPU** | 2 |

| **RAM** | 4 Go |

| **Disque** | 30 Go |

| **Rôle** | Capture et analyse du trafic réseau |

Détails :

- Interface réseau configurée en **mode promiscuous**
- Snort 3 écoute sur `ens33` (ou `eth0`)
- Logs stockés dans `/var/log/snort/`
- Alertes dans `/var/log/snort/alert_fast.txt`

---

kali-attacker — Machine attaquante

| Propriété | Valeur |

|---|---|

| **OS** | Kali Linux 2024.1 |

| **IP** | 192.168.10.20/24 |

| **vCPU** | 2 |

| **RAM** | 2 Go |

| **Disque** | 20 Go |

| **Rôle** | Simulation d'attaques réseau |

Outils utilisés :

- `nmap` — scan de ports et découverte réseau
- `hydra` — brute-force SSH/FTP
- `hping3` — ping flood et forge de paquets
- `metasploit` — exploitation de vulnérabilités

---

ubuntu-target — Serveur cible Linux

| Propriété | Valeur |

|---|---|

| **OS** | Ubuntu Server 22.04 LTS |

| **IP** | 192.168.10.30/24 |

| **vCPU** | 1 |

| **RAM** | 2 Go |

| **Disque** | 20 Go |

| **Rôle** | Cible principale des attaques simulées |

Services exposés :

- SSH sur le port **22**
- Apache2 (HTTP) sur le port **80**
- vsftpd (FTP) sur le port **21**

---

win-server — Serveur cible Windows

| Propriété | Valeur |

|---|---|

| **OS** | Windows Server 2022 |

| **IP** | 192.168.10.40/24 |

| **vCPU** | 2 |

| **RAM** | 4 Go |

| **Disque** | 60 Go |

| **Rôle** | Cible secondaire (services Windows) |

Services exposés :

- SMB sur le port **445**
- RDP sur le port **3389**

---

analyst-ws — Poste analyste

| Propriété | Valeur |

|---|---|

| **OS** | Windows 10 Pro |

| **IP** | 192.168.10.50/24 |

| **vCPU** | 2 |

| **RAM** | 4 Go |

| **Disque** | 40 Go |

| **Rôle** | Consultation des alertes et logs |

Usage :

- Connexion SSH vers `snort-ids` pour lire les alertes
- Éventuellement : interface web Snorby ou Splunk

🔗 Navigation

← Retour : 02 - Objectifs du projet

→ Suite : 04 - Tableau infrastructure

---

tags:

- BTS-SIO
- SISR
- E6
- Snort
- Infrastructure

parents: "00 - Index du projet"

---

04 – Tableau de l'infrastructure

📋 Récapitulatif des machines virtuelles

| Nom de la machine | Système d'exploitation | Adresse IP | Masque | Rôle |

|---|---|---|---|---|

| `snort-ids` | Ubuntu Server 22.04 LTS | 192.168.10.10 | /24 | Serveur IDS Snort (écoute passive) |

| `kali-attacker` | Kali Linux 2024.1 | 192.168.10.20 | /24 | Machine attaquante (tests d'intrusion) |

| `ubuntu-target` | Ubuntu Server 22.04 LTS | 192.168.10.30 | /24 | Serveur cible Linux (Apache, SSH, FTP) |

| `win-server` | Windows Server 2022 | 192.168.10.40 | /24 | Serveur cible secondaire (AD, SMB, RDP) |

| `analyst-ws` | Windows 10 Pro | 192.168.10.50 | /24 | Poste analyste (consultation des alertes) |

🌐 Paramètres réseau communs

| Paramètre | Valeur |

|---|---|

| **Réseau** | 192.168.10.0/24 |

| **Masque** | 255.255.255.0 |

| **Passerelle** | 192.168.10.1 (interface hôte) |

| **DNS** | 8.8.8.8 (si accès Internet requis) |

| **Type de réseau** | Host-only (VMware) / Internal Network (VirtualBox) / vmbr1 (Proxmox) |

🔌 Ports et services exposés

| Machine | Port | Service | Protocole |

|---|---|---|---|

| `ubuntu-target` | 22 | SSH | TCP |

| `ubuntu-target` | 80 | HTTP (Apache2) | TCP |

| `ubuntu-target` | 21 | FTP (vsftpd) | TCP |

| `win-server` | 445 | SMB | TCP |

| `win-server` | 3389 | RDP | TCP |

| `snort-ids` | 22 | SSH (administration) | TCP |

⚠ Sécurité du labo

> Ces services sont intentionnellement exposés pour les besoins du test. En production, l'exposition de SSH, FTP et RDP sans authentification forte serait inacceptable.

💾 Ressources totales nécessaires

| Ressource | Total requis |

|---|---|

| **vCPU** | 9 cœurs |

| **RAM** | 16 Go |

| **Stockage** | ~170 Go |

🔥 Configuration minimale recommandée

> Machine hôte : 16 Go RAM, 200 Go SSD, processeur Intel/AMD 4 cœurs avec virtualisation activée (VT-x / AMD-V dans le BIOS).

🔗 Navigation

← Retour : 03 - Architecture technique

→ Suite : 05 - Schéma réseau

---

tags:

- BTS-SIO
- SISR
- E6
- Snort
- Schéma
- Réseau

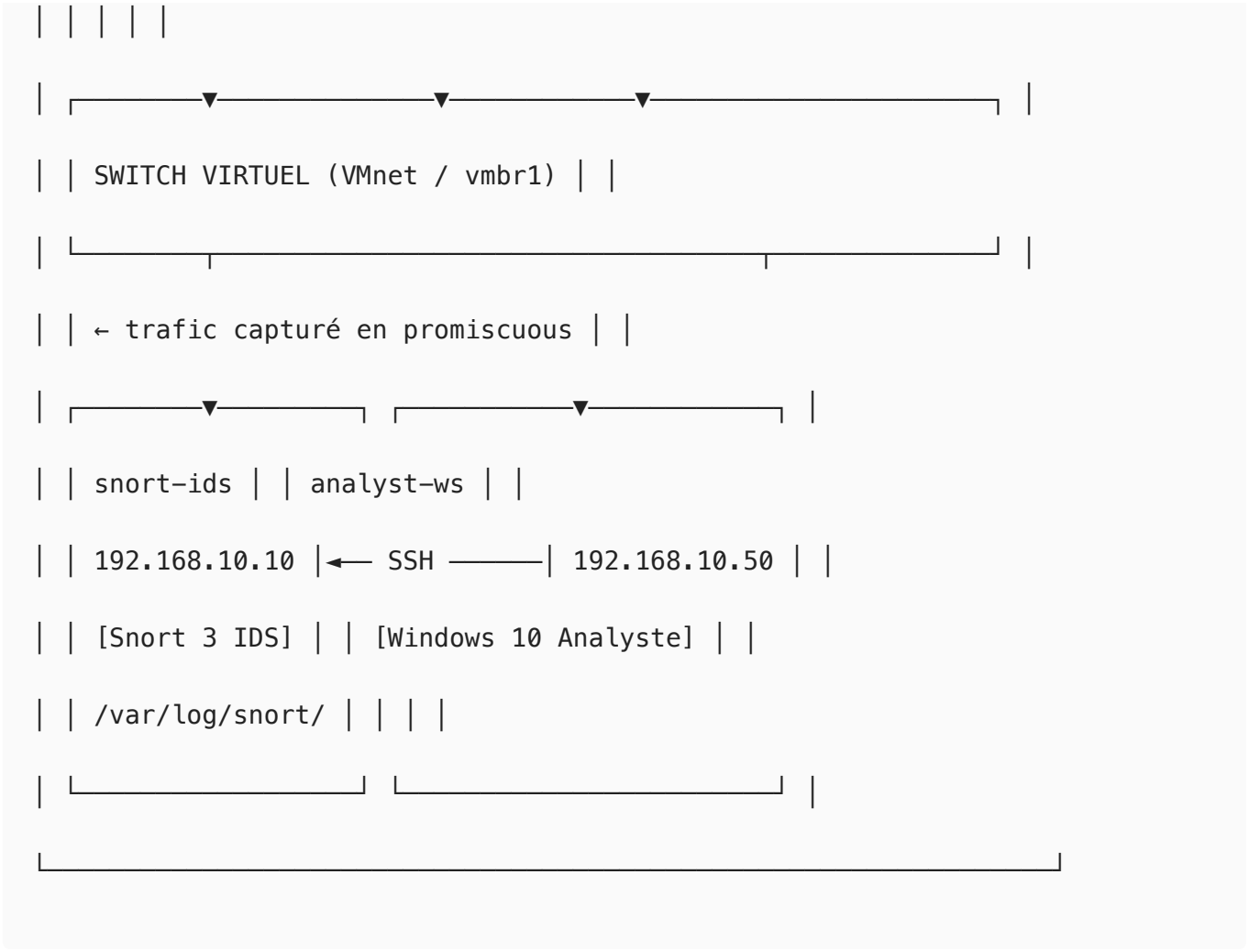
parents: "00 - Index du projet"

---

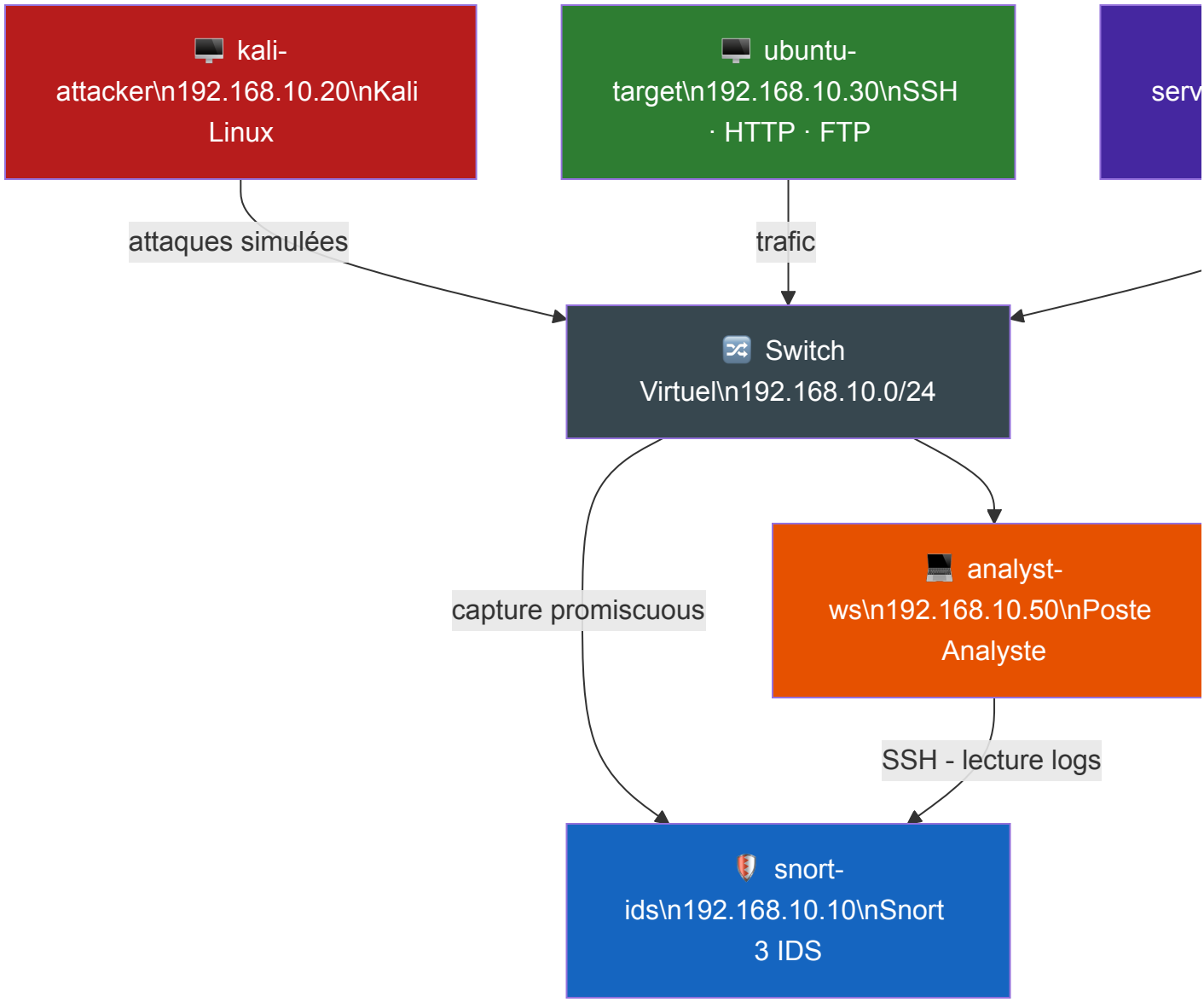
05 – Schéma d'architecture réseau

🗺️ Topologie – Vue ASCII

```
┌──────────────────────────────────────────────────────────────────────┐
│ RÉSEAU VIRTUEL : 192.168.10.0/24 │
│ │
│ ┌──────────────────┐ ┌──────────────────┐ │
│ │ kali-attacker │ │ ubuntu-target │ │
│ │ 192.168.10.20 │ │ 192.168.10.30 │ │
│ │ [Kali Linux] │ │ [SSH/HTTP/FTP] │ │
│ └────────┬─────────┘ └────────┬─────────┘ │
│ │ │ │
│ │ ┌──────────────────┐ │ │
│ │ win-server │ │ │
│ │ 192.168.10.40 │ │ │
│ │ [SMB / RDP] │ │ │
│ │ └────────┬─────────┘ │ │
```



📊 Diagramme Mermaid (pour draw.io / Obsidian)



📝 Flux réseau documentés

| Source | Destination | Port | Description |

|---|---|---|---|

| kali-attacker | ubuntu-target | 22 | Brute-force SSH |

| kali-attacker | ubuntu-target | 80 | Requêtes HTTP |

| kali-attacker | ubuntu-target | 21 | Brute-force FTP |

| kali-attacker | win-server | 445 | Scan SMB |

| kali-attacker | Tout le réseau | * | Scan Nmap |

| kali-attacker | ubuntu-target | * | hping3 flood ICMP |

| analyst-ws | snort-ids | 22 | Administration SSH |

🔧 Instructions pour reproduire dans draw.io

1. Ouvrir draw.io ou l'app desktop
2. Créer un cadre représentant le réseau `192.168.10.0/24`
3. Placer 5 machines virtuelles avec les icônes "Server" ou "Computer"
4. Relier les machines à un switch central avec des flèches
5. Colorier `snort-ids` en bleu, `kali-attacker` en rouge
6. Ajouter les IP sous chaque machine
7. Indiquer les flux d'attaques avec des flèches rouges pointillées
8. Indiquer la capture promiscuous avec une flèche bleue vers Snort

---

🔗 Navigation

← Retour : 04 - Tableau infrastructure

→ Suite : 06 - Installation Snort

---

tags:

- BTS-SIO
- SISR
- E6
- Snort
- Installation
- Linux

parents: "00 - Index du projet"

---

06 – Installation de Snort

ⓘ Machine concernée

> Toutes les commandes s'exécutent sur **snort-ids** (192.168.10.10) en tant que root ou avec `sudo`.

📦 Étape 1 – Mise à jour du système

```
sudo apt update && sudo apt upgrade -y

sudo apt install -y curl wget git
```

📦 Étape 2 – Installation des dépendances

```
sudo apt install -y \

build-essential \

libpcap-dev \

libpcre3-dev \

libdumbnet-dev \

bison flex \

zlib1g-dev \

liblzma-dev \

openssl libssl-dev \

libnghttp2-dev \

libdnet \

cmake \
```

```
libhwloc-dev \
pkg-config \
luajit libluajit-5.1-dev
```

📦 Étape 3 – Installation de libdaq (dépendance Snort 3)

```
cd /tmp
git clone https://github.com/snort3/libdaq.git
cd libdaq

./bootstrap
./configure
make
sudo make install
sudo ldconfig
```

📦 Étape 4 – Téléchargement et compilation de Snort 3

```
cd /tmp
wget https://github.com/snort3/snort3/archive/refs/tags/3.3.7.0.tar.gz
tar -xzf 3.3.7.0.tar.gz
cd snort3-3.3.7.0
```

```
# Configuration et compilation
./configure_cmake.sh --prefix=/usr/local --enable-tcmalloc
cd build
make -j$(nproc)
sudo make install
sudo ldconfig
```

✅ Étape 5 – Vérification de l'installation

```
snort --version
```

Résultat attendu :

```
,,_ -*> Snort++ <*-
o" )~ Version 3.3.7.0
'''' By Martin Roesch & The Snort Team
```

🌐 Étape 6 – Téléchargement des règles communautaires

```
# Créer l'arborescence des règles
sudo mkdir -p /etc/snort/rules
```

```
sudo mkdir -p /var/log/snort


# Télécharger les règles Community (gratuites)
cd /tmp
wget 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 /etc/snort/rules/


# Créer le fichier de règles personnalisées
sudo touch /etc/snort/rules/local.rules
```

🔥 Règles Talos (payantes mais plus complètes)

> Pour un usage professionnel, les règles Talos de Snort.org offrent une bien meilleure couverture. Inscription gratuite sur snort.org pour l'accès aux règles Community.

📁 Arborescence finale

```
/etc/snort/
├── snort.conf (ou snort.lua pour Snort 3)
├── rules/
│ ├── local.rules ← nos règles personnalisées
│ └── snort3-community.rules
/var/log/snort/
├── alert_fast.txt ← alertes générées
```

```
├── snort.log.XXXXXXXX ← captures pcap
```

⚠️ Option alternative – Snort 2.9 (plus simple, dépôts Ubuntu)

Si la compilation de Snort 3 est trop complexe, installer Snort 2.9 depuis les dépôts :

```
sudo apt install -y snort

# L'installeur demande :

# - Interface réseau : ens33

# - Réseau local : 192.168.10.0/24
```

⚠ Snort 2 vs Snort 3

> Snort 2 utilise `snort.conf` (ancienne syntaxe). Snort 3 utilise `snort.lua` (Lua). Les deux approches sont valides pour un projet BTS — **Snort 3 est recommandé** car c'est la version maintenue activement.

🔗 Navigation

← Retour : 05 - Schéma réseau

→ Suite : 07 - Configuration Snort

---

tags:

- BTS-SIO
- SISR
- E6
- Snort

- Configuration
- Linux

parents: "00 - Index du projet"

---

07 – Configuration de Snort

🌐 Étape 1 – Configuration de l'interface en mode promiscuous

```
# Identifier l'interface réseau

ip a

# → noter le nom de l'interface, ex : ens33 ou eth0



# Passer l'interface en mode promiscuous

sudo ip link set ens33 promisc on



# Vérification

ip link show ens33

# La ligne doit contenir : PROMISC
```

Pour rendre le mode promiscuous **permanent au démarrage** :

```
sudo nano /etc/systemd/network/10-promisc.network
```

```
[Match]
```

```
Name=ens33


[Link]

Promiscuous=yes
```

```
sudo systemctl restart systemd-networkd
```

⚙️ Étape 2 – Configuration de snort.lua (Snort 3)

Éditer le fichier de configuration principal :

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

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

Contenu de `/etc/snort/snort.lua`

```
-- ============================================

-- snort.lua – Configuration Snort 3 (IDS mode)

-- Projet BTS SIO SISR – TechNova Solutions

-- ============================================


-- Réseau local à surveiller

HOME_NET = '192.168.10.0/24'

EXTERNAL_NET = '!$HOME_NET'
```

```
-- Chemins des règles
RULE_PATH = '/etc/snort/rules'
BUILTIN_RULE_PATH = '/usr/local/lib/snort/plugins'
PLUGIN_RULE_PATH = '/usr/local/lib/snort/plugins'

-- Variables de services
HTTP_PORTS = '80'
SHELLCODE_PORTS = '!80'
ORACLE_PORTS = '1521'
SSH_PORTS = '22'
FTP_PORTS = '21'
SIP_PORTS = '5060:5061'
FILE_DATA_PORTS = '[ [ $HTTP_PORTS ]]'

-- Configuration de la détection
detect = {
rules = RULE_PATH .. '/local.rules,' ..
RULE_PATH .. '/snort3-community.rules',
}

-- Alertes – format rapide (lisible)
alert_fast = {
```

```
file = true,

packet = false,

limit = 10,

}


-- Journalisation des paquets

log_pcap = {

limit = 100,

}


-- Gestion des flux

stream = {}

stream_tcp = {}

stream_udp = {}

stream_icmp = {}


-- Normalisation IP

normalizer = {}


-- Préprocesseurs

arp_spoof = {}

back_orifice = {}

dns = { ports = '53' }
```

```
ftp_telnet = {}

http_inspect = {}

imap = {}

pop = {}

smtp = {}

ssh = { server_ports = '22' }


-- Binder (liaison règles / flux)

binder = {

{ when = { proto = 'tcp', ports = '80' }, use = { type = 'http_inspect' } },

{ when = { proto = 'tcp', ports = '22' }, use = { type = 'ssh' } },

}
```

⚙️ Étape 3 – Test de la configuration

```
# Vérifier la syntaxe de la configuration

sudo snort -c /etc/snort/snort.lua --daq-dir /usr/local/lib/daq -T


# Résultat attendu :

# Snort successfully validated the configuration (with 0 warnings).
```

🚀 Étape 4 – Démarrer Snort en mode IDS

```
# Démarrage manuel (mode console, affichage des alertes)
sudo snort \
-c /etc/snort/snort.lua \
-i ens33 \
-A alert_fast \
-l /var/log/snort \
--daq-dir /usr/local/lib/daq

# Démarrage en arrière-plan (daemon)
sudo snort \
-c /etc/snort/snort.lua \
-i ens33 \
-A alert_fast \
-l /var/log/snort \
--daq-dir /usr/local/lib/daq \
-D
```

| Option | Description |

|---|---|

| `-c` | Fichier de configuration |

| `-i ens33` | Interface d'écoute |

| `-A alert_fast` | Format d'alerte rapide |

| `-l /var/log/snort` | Répertoire des logs |

| `-D` | Mode daemon (arrière-plan) |

| `-T` | Test de configuration uniquement |

🔄 Étape 5 – Créer un service systemd (démarrage automatique)

```
sudo nano /etc/systemd/system/snort3.service
```

```
[Unit]

Description=Snort 3 IDS Service

After=network.target



[Service]

Type=simple

ExecStart=/usr/local/bin/snort \

-c /etc/snort/snort.lua \

-i ens33 \

-A alert_fast \

-l /var/log/snort \

--daq-dir /usr/local/lib/daq

Restart=on-failure

RestartSec=5s

StandardOutput=journal
```

```
StandardError=journal


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

```
sudo systemctl daemon-reload
sudo systemctl enable snort3
sudo systemctl start snort3

# Vérification du statut
sudo systemctl status snort3
```

✅ Vérification finale

```
# Snort tourne-t-il ?
ps aux | grep snort

# Les alertes se génèrent-elles ?
sudo tail -f /var/log/snort/alert_fast.txt

# Envoyer un ping test depuis la machine hôte
```

```
ping 192.168.10.30
```

🔗 Navigation

← Retour : 06 - Installation Snort

→ Suite : 08 - Règles Snort personnalisées

---

tags:

- BTS-SIO
- SISR
- E6
- Snort
- Règles
- IDS

parents: "00 - Index du projet"

---

08 – Règles Snort personnalisées

> ⓘ **Fichier cible**

> Toutes les règles personnalisées sont à placer dans `/etc/snort/rules/local.rules`

```
```

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

```
```

📐 Anatomie d'une règle Snort

```
action proto src_ip src_port direction dst_ip dst_port (options)
| | | | | | |
alert tcp any any -> $HOME_NET 22 (msg:"SSH brute force"; ...)
```

Champ	Description	Valeurs possibles
action	Que faire si la règle correspond	`alert`, `log`, `drop`, `pass`
proto	Protocole réseau	`tcp`, `udp`, `icmp`, `ip`
src_ip	IP source	adresse, `any`, `$EXTERNAL_NET`
src_port	Port source	numéro, `any`, plage `1:1024`
direction	Sens du flux	`->` (unidirectionnel), `<>` (bidirectionnel)
dst_ip	IP destination	adresse, `any`, `$HOME_NET`
dst_port	Port destination	numéro, `any`
options	Paramètres de la règle	`msg`, `sid`, `rev`, `threshold` ...

🔴 Règle 1 – Détection de scan ICMP (ping)

```
# Détection d'un ping simple
alert icmp any any -> $HOME_NET any (
msg:"ICMP Ping détecté";
itype:8;
sid:1000001;
rev:1;
```

```
)
```

```
# Détection d'un ping flood (> 5 pings en 1 seconde depuis la même source)
alert icmp any any -> $HOME_NET any (
msg:"ICMP Ping Flood détecté - possible DoS";
itype:8;
threshold: type both, track by_src, count 5, seconds 1;
sid:1000002;
rev:1;
)
```

🔴 Règle 2 – Détection de scan de ports (Nmap)

```
# Détection d'un SYN scan (Nmap -sS)
alert tcp any any -> $HOME_NET any (
msg:"SYN Scan détecté - possible scan Nmap";
flags:S,12;
threshold: type both, track by_src, count 15, seconds 3;
sid:1000003;
rev:1;
)
```

```
# Détection d'un scan NULL (Nmap -sN)
alert tcp any any -> $HOME_NET any (
msg:"TCP NULL Scan détecté";
flags:0;
sid:1000004;
rev:1;
)
```

```
# Détection d'un scan XMAS (Nmap -sX)
alert tcp any any -> $HOME_NET any (
msg:"TCP XMAS Scan détecté - flags FPU";
flags:FPU;
sid:1000005;
rev:1;
)
```

```
# Détection de scan UDP (nombreuses connexions refusées)
alert udp any any -> $HOME_NET any (
msg:"UDP Scan détecté - possible Nmap -sU";
threshold: type both, track by_src, count 20, seconds 5;
sid:1000006;
rev:1;
```

```
)
```

🔴 Règle 3 – Tentative de brute-force SSH

```
# Détection de brute-force SSH (> 5 connexions en 10 secondes)

alert tcp any any -> $HOME_NET 22 (

msg:"BRUTE FORCE SSH détecté - tentatives multiples";

flags:S,12;

threshold: type both, track by_src, count 5, seconds 10;

sid:1000007;

rev:1;

)
```

```
# Détection de connexion SSH depuis l'extérieur du réseau local

alert tcp $EXTERNAL_NET any -> $HOME_NET 22 (

msg:"Connexion SSH depuis IP externe";

flags:S;

sid:1000008;

rev:1;

)
```

🔴 Règle 4 – Tentative de brute-force FTP

```
# Brute-force FTP (> 6 tentatives en 10 secondes)

alert tcp any any -> $HOME_NET 21 (

msg:"BRUTE FORCE FTP détecté";

flags:S,12;

threshold: type both, track by_src, count 6, seconds 10;

sid:1000009;

rev:1;

)
```

```
# Tentative de connexion FTP anonyme

alert tcp any any -> $HOME_NET 21 (

msg:"Connexion FTP anonyme tentée";

content:"USER anonymous";

nocase;

sid:1000010;

rev:1;

)
```

🟡 Règle 5 – Trafic HTTP suspect

```
# Tentative de traversée de répertoire (directory traversal)

alert tcp any any -> $HOME_NET 80 (

msg:"Directory Traversal détecté - ../";

content:"../";

http_uri;

sid:1000011;

rev:1;

)
```

```
# Tentative d'injection SQL basique

alert tcp any any -> $HOME_NET 80 (

msg:"Possible injection SQL détectée";

content:"' OR '1'='1";

nocase;

http_uri;

sid:1000012;

rev:1;

)
```

```
# Tentative de scan Nikto (user-agent)

alert tcp any any -> $HOME_NET 80 (

msg:"Scanner web Nikto détecté";
```

```
content:"Nikto";

http_header;

nocase;

sid:1000013;

rev:1;

)
```

🟡 Règle 6 – Accès SMB suspect

```
# Connexion SMB depuis une source non autorisée

alert tcp any any -> $HOME_NET 445 (

msg:"Connexion SMB suspecte détectée";

flags:S;

threshold: type both, track by_src, count 3, seconds 5;

sid:1000014;

rev:1;

)
```

🟢 Règle 7 – Trafic ICMP de type "unreachable" anormal

```
# ICMP Destination Unreachable en masse (possible scan)

alert icmp any any -> $HOME_NET any (
```

```
msg:"ICMP Destination Unreachable en masse";

itype:3;

threshold: type both, track by_src, count 10, seconds 2;

sid:1000015;

rev:1;

)
```

📋 Récapitulatif des règles

| SID | Nom | Protocole | Priorité |

|---|---|---|---|

| 1000001 | ICMP Ping simple | ICMP | 🟢 Basse |

| 1000002 | ICMP Ping flood | ICMP | 🔴 Haute |

| 1000003 | SYN Scan (Nmap) | TCP | 🔴 Haute |

| 1000004 | NULL Scan | TCP | 🔴 Haute |

| 1000005 | XMAS Scan | TCP | 🔴 Haute |

| 1000006 | UDP Scan | UDP | 🟡 Moyenne |

| 1000007 | Brute-force SSH | TCP | 🔴 Haute |

| 1000008 | SSH externe | TCP | 🟡 Moyenne |

| 1000009 | Brute-force FTP | TCP | 🔴 Haute |

| 1000010 | FTP anonyme | TCP | 🟡 Moyenne |

| 1000011 | Directory traversal | TCP | 🔴 Haute |

| 1000012 | SQLi basique | TCP | 🔴 Haute |

| 1000013 | Scanner Nikto | TCP | 🟡 Moyenne |

| 1000014 | SMB suspect | TCP | 🟡 Moyenne |

| 1000015 | ICMP Unreachable flood | ICMP | 🟢 Basse |

🔄 Recharger Snort après modification des règles

```
# Tester la syntaxe des règles avant rechargement

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


# Redémarrer le service

sudo systemctl restart snort3
```

🔗 Navigation

← Retour : 07 - Configuration Snort

→ Suite : 09 - Simulation d'attaques

---

tags:

- BTS-SIO
- SISR
- E6
- Snort
- Kali
- Attaques
- Tests

parents: "00 - Index du projet"

---

09 – Simulation d'attaques

⚠ **Avertissement légal**

> Ces attaques sont réalisées **exclusivement dans le cadre du laboratoire virtualisé isolé**. Il est **illégal** d'utiliser ces techniques sur des systèmes sans autorisation explicite.

ⓘ **Machine utilisée**

> Toutes les commandes ci-dessous s'exécutent sur **kali-attacker** (192.168.10.20).

🔍 Préparation – Vérifier la connectivité

```
# Depuis kali-attacker, vérifier que les cibles répondent

ping -c 2 192.168.10.30 # ubuntu-target

ping -c 2 192.168.10.40 # win-server

ping -c 2 192.168.10.10 # snort-ids (doit aussi répondre)
```

Depuis **snort-ids**, surveiller les alertes en temps réel :

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

🔴 Attaque 1 – Scan de ports avec Nmap

SYN Scan (furtif)

```
# Scan SYN sur toutes les machines du réseau

sudo nmap -sS 192.168.10.0/24
```

```
# Scan SYN sur la machine cible avec détection OS et services
sudo nmap -sS -sV -O 192.168.10.30


# Scan de ports spécifiques
sudo nmap -sS -p 22,80,21,443,3306 192.168.10.30
```

Scan agressif (toutes techniques)

```
# Scan agressif : OS, versions, scripts, traceroute
sudo nmap -A 192.168.10.30
```

Scan NULL et XMAS (pour tester les règles évasives)

```
# NULL scan
sudo nmap -sN 192.168.10.30


# XMAS scan
sudo nmap -sX 192.168.10.30


# FIN scan
sudo nmap -sF 192.168.10.30
```

Scan UDP

```
# UDP scan (lent mais utile pour tester la règle 1000006)
sudo nmap -sU --top-ports 50 192.168.10.30
```

✓ **Alertes attendues sur Snort**

- SID 1000003 : SYN Scan détecté
- SID 1000004 : NULL Scan détecté
- SID 1000005 : XMAS Scan détecté
- SID 1000006 : UDP Scan détecté

🔴 Attaque 2 – Brute-force SSH avec Hydra

Préparation – Créer une wordlist

```
# Utiliser la wordlist intégrée rockyou
ls /usr/share/wordlists/rockyou.txt

# Si compressée, décompresser
sudo gunzip /usr/share/wordlists/rockyou.txt.gz

# Ou créer une mini wordlist pour le test
cat > /tmp/passwords.txt << EOF
admin
password
123456
```

```
root

toor

qwerty

letmein

P@ssw0rd

EOF
```

Brute-force SSH

```
# Brute-force SSH sur ubuntu-target

hydra -l root -P /tmp/passwords.txt 192.168.10.30 ssh -t 4 -V


# Avec plusieurs noms d'utilisateur

hydra -L /usr/share/wordlists/metasploit/unix_users.txt \

-P /tmp/passwords.txt \

192.168.10.30 ssh -t 4


# Options clés :

# -l : login unique

# -L : fichier de logins

# -P : fichier de mots de passe

# -t : nombre de threads

# -V : mode verbose (affiche chaque tentative)
```

✓ **Alertes attendues sur Snort**

- SID 1000007 : BRUTE FORCE SSH détecté

Brute-force FTP

```
# Brute-force FTP sur ubuntu-target

hydra -l anonymous -P /tmp/passwords.txt 192.168.10.30 ftp -V


# Avec login admin

hydra -l ftpuser -P /usr/share/wordlists/rockyou.txt 192.168.10.30 ftp
```

✓ **Alertes attendues**

- SID 1000009 : BRUTE FORCE FTP détecté
- SID 1000010 : Connexion FTP anonyme tentée

🔴 Attaque 3 – Ping flood avec hping3

Ping flood ICMP classique

```
# Ping flood simple (1 000 paquets ICMP)

sudo hping3 -1 --flood 192.168.10.30


# Ping flood avec taille de paquet personnalisée

sudo hping3 -1 -d 1000 --flood 192.168.10.30
```

```
# Avec compteur limité (100 paquets, pour ne pas saturer le réseau du labo)
sudo hping3 -1 -c 100 -i u10000 192.168.10.30
# -c 100 : 100 paquets
# -i u10000 : intervalle de 10 ms entre chaque paquet
```

SYN flood (simulation DoS TCP)

```
# SYN flood sur le port 80 d'ubuntu-target
sudo hping3 -S --flood -V -p 80 192.168.10.30

# SYN flood avec IP source aléatoire (usurpation)
sudo hping3 -S --flood -V --rand-source -p 22 192.168.10.30
```

✓ **Alertes attendues**

- SID 1000002 : ICMP Ping Flood détecté
- SID 1000003 : SYN Scan (sur le flood TCP)

🔴 Attaque 4 – Exploitation web (pour tester les règles HTTP)

```
# Scan web avec Nikto (déclenche la règle 1000013)
nikto -h http://192.168.10.30

# Test de directory traversal manuel
curl "http://192.168.10.30/../../../etc/passwd"
```

```
# Test d'injection SQL dans l'URL
curl "http://192.168.10.30/index.php?id=1' OR '1'='1"
```

✓ **Alertes attendues**

- SID 1000011 : Directory Traversal détecté
- SID 1000012 : Possible injection SQL détectée
- SID 1000013 : Scanner web Nikto détecté

📋 Tableau de synthèse des tests

#	Outil	Commande principale	Règle déclenchée	SID
1	Nmap	`nmap -sS 192.168.10.0/24`	SYN Scan	1000003
2	Nmap	`nmap -sN 192.168.10.30`	NULL Scan	1000004
3	Nmap	`nmap -sX 192.168.10.30`	XMAS Scan	1000005
4	Nmap	`nmap -sU 192.168.10.30`	UDP Scan	1000006
5	Hydra	`hydra -l root -P ... ssh`	Brute-force SSH	1000007
6	Hydra	`hydra -l anon -P ... ftp`	Brute-force FTP	1000009
7	hping3	`hping3 -1 --flood`	ICMP Flood	1000002
8	hping3	`hping3 -S --flood -p 80`	SYN Flood	1000003
9	Nikto	`nikto -h http://...`	Scanner Nikto	1000013
10	curl	`curl "...?id=1' OR '1'='1"`	SQLi	1000012

🔗 Navigation

← Retour : 08 - Règles Snort personnalisées

→ Suite : 10 - Analyse des alertes

---

tags:

- BTS-SIO
- SISR
- E6
- Snort
- Alertes
- Logs
- Analyse

parents: "00 - Index du projet"

---

10 – Analyse des alertes et logs

📂 Localisation des fichiers de logs

```
ls -lh /var/log/snort/

# alert_fast.txt ← alertes en format texte lisible

# snort.log.XXXXXX ← capture pcap brute
```

📖 Format d'une alerte Snort (alert_fast)

Exemple d'alerte générée par un scan Nmap :

```
03/15-14:32:01.123456 [**] [1:1000003:1] SYN Scan détecté – possible scan
```

```
Nmap [**] [Priority: 0] {TCP} 192.168.10.20:54321 -> 192.168.10.30:22
```

Décomposition du format

| Champ | Valeur dans l'exemple | Signification |

|---|---|---|

| `03/15-14:32:01.123456` | Date et heure | Timestamp de la détection |

| `[**]` | Séparateur | Délimiteur Snort |

| `[1:1000003:1]` | GID:SID:REV | Identifiant unique de la règle |

| `SYN Scan détecté...` | Message | Texte du champ `msg:` de la règle |

| `[Priority: 0]` | Priorité | 0=info, 1=haute, 2=moyenne, 3=basse |

| `{TCP}` | Protocole | TCP, UDP, ICMP |

| `192.168.10.20:54321` | IP:Port source | Origine du paquet |

| `->` | Direction | Flux unidirectionnel |

| `192.168.10.30:22` | IP:Port destination | Cible du paquet |

🔍 Commandes de lecture des logs

```
# Voir les alertes en temps réel
sudo tail -f /var/log/snort/alert_fast.txt


# Voir les 20 dernières alertes
sudo tail -20 /var/log/snort/alert_fast.txt


# Filtrer par IP source (ex: la machine kali)
grep "192.168.10.20" /var/log/snort/alert_fast.txt
```

```
# Filtrer par type d'alerte (ex: SSH)

grep "SSH" /var/log/snort/alert_fast.txt



# Compter les alertes par type

grep -o '\[1:[0-9]*' /var/log/snort/alert_fast.txt | sort | uniq -c | sort -
rn



# Filtrer par plage horaire

grep "03/15-14:3" /var/log/snort/alert_fast.txt
```

📊 Exemple de session d'analyse complète

1 – Démarrer la surveillance en temps réel

```
# Terminal 1 (snort-ids) : surveiller les alertes

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



# Terminal 2 (kali-attacker) : lancer l'attaque

sudo nmap -sS 192.168.10.30
```

2 – Résultat observé dans les logs

```
03/15-14:32:01.100001 [**] [1:1000003:1] SYN Scan détecté - possible scan
Nmap [**] {TCP} 192.168.10.20:45231 -> 192.168.10.30:22
```

```
03/15-14:32:01.100022 [**] [1:1000003:1] SYN Scan détecté - possible scan
Nmap [**] {TCP} 192.168.10.20:45231 -> 192.168.10.30:80

03/15-14:32:01.100045 [**] [1:1000003:1] SYN Scan détecté - possible scan
Nmap [**] {TCP} 192.168.10.20:45231 -> 192.168.10.30:21

03/15-14:32:01.200001 [**] [1:1000002:1] ICMP Ping Flood détecté [**] {ICMP}
192.168.10.20 -> 192.168.10.30
```

3 – Analyse et interprétation

Interprétation de l'alerte SYN Scan

- **Source** : 192.168.10.20 (kali-attacker) → IP de la machine attaquante identifiée
- **Destination** : 192.168.10.30 (ubuntu-target) → cible de l'attaque
- **Comportement** : plusieurs connexions TCP SYN vers des ports différents en très peu de temps → caractéristique d'un scan de ports
- **Règle déclenchée** : SID 1000003 → notre règle personnalisée

---

📁 Analyse des captures pcap (Wireshark)

```
# Analyser le fichier pcap avec tcpdump

sudo tcpdump -r /var/log/snort/snort.log.XXXXXXX



# Filtrer sur un protocole

sudo tcpdump -r /var/log/snort/snort.log.XXXXXXX tcp



# Filtrer sur une IP source

sudo tcpdump -r /var/log/snort/snort.log.XXXXXXX src 192.168.10.20
```

```
# Exporter vers Wireshark (depuis analyst-ws)
scp matthis@192.168.10.10:/var/log/snort/snort.log.* .
# Puis ouvrir dans Wireshark
```

📊 Statistiques des alertes

```
# Compter le total d'alertes
wc -l /var/log/snort/alert_fast.txt

# Top 10 des IPs sources les plus actives
grep -oP '\d+\.\d+\.\d+\.\d+(?=:\d+ ->)' /var/log/snort/alert_fast.txt \
| sort | uniq -c | sort -rn | head -10

# Top 10 des règles les plus déclenchées
grep -oP '\[1:\d+:\d+\]' /var/log/snort/alert_fast.txt \
| sort | uniq -c | sort -rn | head -10

# Alertes par heure
grep -oP '\d{2}/\d{2}-\d{2}' /var/log/snort/alert_fast.txt \
| sort | uniq -c
```

📋 Tableau d'interprétation des alertes

| Alerte observée | Signification probable | Action recommandée |

|---|---|---|

| Nombreux SYN sur ports variés | Scan de ports en cours | Bloquer l'IP source, investiguer |

| Alertes SSH répétées depuis une IP | Brute-force en cours | Bannir l'IP (fail2ban), alerter l'admin |

| ICMP flood | Déni de service ou test réseau | Vérifier si source interne, rate-limiting |

| Directory traversal HTTP | Tentative d'exploitation web | Patcher l'appli, WAF, alerter le dev |

| FTP anonyme | Accès non autorisé tenté | Désactiver FTP anonyme, vérifier config |

🔄 Rotation et archivage des logs

```
# Configurer logrotate pour Snort

sudo nano /etc/logrotate.d/snort
```

```
/var/log/snort/alert_fast.txt {

daily

rotate 30

compress

missingok

notifempty

postrotate

systemctl restart snort3

endscript
```

```
}
```

🔗 Navigation

← Retour : 09 - Simulation d'attaques

→ Suite : 11 - Sécurisation et améliorations

---

tags:

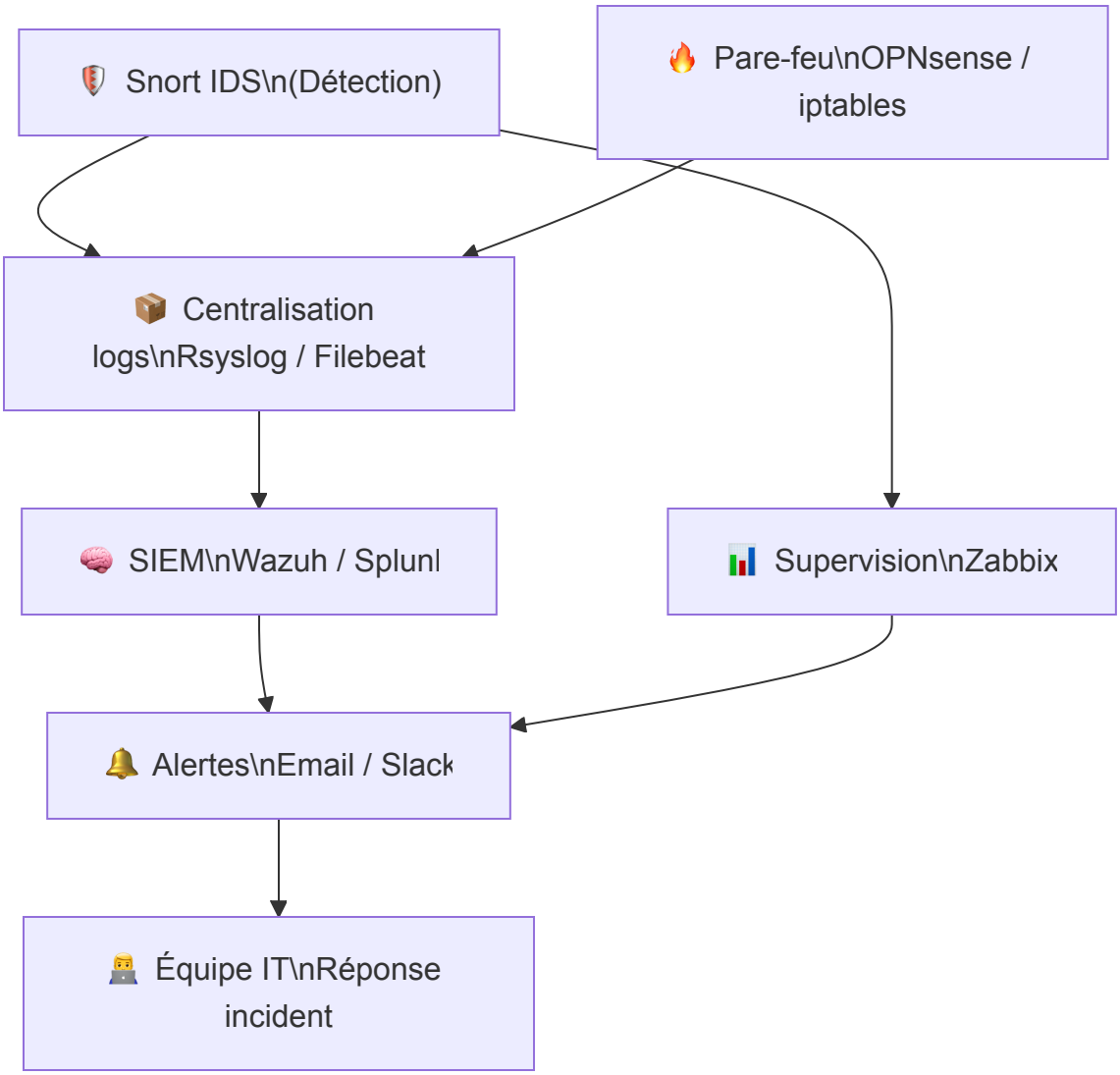
- BTS-SIO
- SISR
- E6
- Snort
- Sécurité
- SIEM
- Améliorations

parents: "00 - Index du projet"

---

11 – Sécurisation et améliorations possibles

🗺️ Vision d'ensemble de l'écosystème cible



1️⃣ Ajout d'un pare-feu

Avec iptables (sur ubuntu-target)

```
# Bloquer une IP détectée comme malveillante par Snort

sudo iptables -A INPUT -s 192.168.10.20 -j DROP


# Limiter les connexions SSH (anti brute-force)

sudo iptables -A INPUT -p tcp --dport 22 -m state --state NEW \

-m recent --set --name SSH
```

```
sudo iptables -A INPUT -p tcp --dport 22 -m state --state NEW \
-m recent --update --seconds 60 --hitcount 5 --name SSH -j DROP

# Sauvegarder les règles
sudo iptables-save > /etc/iptables/rules.v4
```

Avec OPNsense (recommandé pour le projet)

> 🔥 **Intégration OPNsense + Snort**

> OPNsense possède un plugin Snort natif qui permet de faire passer Snort en mode **IPS (Intrusion Prevention System)** : les paquets malveillants sont **bloqués** automatiquement au niveau du pare-feu, sans modification manuelle d'iptables.

- Plugin : `os-snort` dans OPNsense
- Mode recommandé : **Inline IPS** (bloquant)

2️⃣ Centralisation des logs (Syslog)

Configurer Snort pour envoyer vers rsyslog

```
# Sur snort-ids, configurer rsyslog
sudo nano /etc/rsyslog.conf
```

```
# Forwarder les logs Snort vers un serveur central
*.* @192.168.10.50:514 # UDP
```

```
*.* @@192.168.10.50:514 # TCP (plus fiable)
```

```
sudo systemctl restart rsyslog
```

Configurer Filebeat pour ELK Stack

```
# Installation de Filebeat

wget https://artifacts.elastic.co/downloads/beats/filebeat/filebeat-8.x-amd64.deb

sudo dpkg -i filebeat-8.x-amd64.deb



# Configuration pour lire les alertes Snort

sudo nano /etc/filebeat/filebeat.yml
```

```
filebeat.inputs:

- type: log

enabled: true

paths:

- /var/log/snort/alert_fast.txt

fields:

source: snort

type: ids_alert
```

```
output.logstash:

hosts: ["192.168.10.50:5044"]
```

3️⃣ Intégration avec un SIEM

Option A – Wazuh (open-source, recommandé)

> ⓘ **Wazuh**

> Wazuh est une plateforme SIEM/XDR open-source qui intègre nativement Snort. Elle offre :
>
> - Corrélation d'événements
> - Tableaux de bord de sécurité
> - Alertes par email/Slack
> - Conformité (PCI-DSS, GDPR)

```
# Installer l'agent Wazuh sur snort-ids

curl -s https://packages.wazuh.com/key/GPG-KEY-WAZUH | sudo apt-key add -

echo "deb https://packages.wazuh.com/4.x/apt/ stable main" | \

sudo tee /etc/apt/sources.list.d/wazuh.list

sudo apt update

sudo apt install -y wazuh-agent


# Configurer l'agent pour pointer vers le serveur Wazuh

sudo nano /var/ossec/etc/ossec.conf

# Modifier <address> avec l'IP du serveur Wazuh
```

Option B – Splunk (interface plus riche)

```
# Configurer Snort pour écrire en format JSON (compatible Splunk)

# Dans snort.lua, ajouter :

# alert_json = { file = true, limit = 100 }



# Ajouter l'add-on Snort dans Splunk

# Splunk Add-on for Snort : disponible sur Splunkbase
```

4️⃣ Supervision avec Zabbix

🔥 Lien avec le projet Zabbix

> Ce projet Snort peut s'intégrer directement avec le projet Zabbix précédent !

Surveiller le service Snort depuis Zabbix

```
# Sur snort-ids, installer l'agent Zabbix

wget https://repo.zabbix.com/zabbix/7.4/ubuntu/pool/main/z/zabbix-release/

sudo apt install -y zabbix-agent2



# Configurer l'agent

sudo nano /etc/zabbix/zabbix_agent2.conf

# Server=192.168.X.X (IP du serveur Zabbix)
```

Items Zabbix personnalisés pour Snort

```
# Vérifier si Snort tourne

sudo nano /etc/zabbix/zabbix_agent2.d/snort.conf
```

```
UserParameter=snort.running, systemctl is-active snort3 | grep -c active

UserParameter=snort.alerts_count, wc -l < /var/log/snort/alert_fast.txt

UserParameter=snort.last_alert, tail -1 /var/log/snort/alert_fast.txt
```

Triggers Zabbix recommandés

| Trigger | Condition | Sévérité |

|---|---|---|

| Snort service down | `snort.running = 0` | 🔴 Disaster |

| Pic d'alertes anormal | `> 50 alertes/min` | 🟡 Warning |

| Espace disque logs faible | `< 2 Go disponibles` | 🟡 Warning |

5️⃣ Évolution vers IPS (mode bloquant)

> 🔗 **IDS vs IPS**
>
> | Mode | Description | Snort |
>
> |---|---|---|
>
> | **IDS** (actuel) | Détecte et alerte, ne bloque pas | Mode passif – écoute sur `ens33` |
>
> | **IPS** (évolution) | Bloque le trafic en temps réel | Mode inline – entre deux interfaces réseau |

```
# Snort en mode IPS inline (nécessite deux interfaces réseau sur snort-ids)
```

```
sudo snort \
-c /etc/snort/snort.lua \
-i ens33:ens34 \ # deux interfaces : entrée:sortie
-A alert_fast \
-l /var/log/snort \
--daq-dir /usr/local/lib/daq \
--daq afpacket \
--daq-var buffer_size_mb=128
```

6️⃣ Fail2ban – Bannissement automatique des IPs

```
sudo apt install -y fail2ban


# Configuration pour bannir les IPs détectées par Snort
sudo nano /etc/fail2ban/jail.local
```

```
[snort-ssh]
enabled = true
port = ssh
filter = sshd
logpath = /var/log/auth.log
maxretry = 5
```

```
bantime = 3600

findtime = 600
```

```
sudo systemctl enable fail2ban

sudo systemctl start fail2ban


# Voir les IPs bannies

sudo fail2ban-client status sshd
```

📋 Tableau des améliorations

| Amélioration | Complexité | Bénéfice | Outils |

|---|---|---|---|

| Pare-feu iptables | 🟢 Faible | Blocage manuel des IPs | iptables |

| OPNsense IPS | 🟡 Moyenne | Blocage automatique en ligne | OPNsense + plugin Snort |

| Centralisation logs | 🟡 Moyenne | Vue unifiée de tous les logs | rsyslog, Filebeat |

| SIEM Wazuh | 🔴 Élevée | Corrélation et dashboards | Wazuh |

| Supervision Zabbix | 🟡 Moyenne | Monitoring du service Snort | Zabbix Agent |

| Fail2ban | 🟢 Faible | Bannissement SSH automatique | fail2ban |

| Mode IPS inline | 🔴 Élevée | Blocage temps réel | Snort 3 + afpacket DAQ |

🔗 Navigation

← Retour : 10 - Analyse des alertes

→ Suite : 12 - Compétences BTS SIO SISR

tags:

- BTS-SIO
- SISR
- E6
- Compétences
- Référentiel

parents: "00 - Index du projet"

---

12 – Compétences BTS SIO SISR mobilisées

📋 Référentiel de compétences

Bloc 1 – Support et mise à disposition de services informatiques

| Compétence | Intitulé | Mise en œuvre dans le projet |

|---|---|---|

| **B1.2** | Réponse aux incidents et aux demandes d'assistance et d'évolution | Analyse des alertes Snort, réponse aux incidents de sécurité simulés |

| **B1.4** | Travail en mode projet | Rédaction du dossier technique, documentation Obsidian, planification |

Bloc 2 – Administration des systèmes et des réseaux

| Compétence | Intitulé | Mise en œuvre dans le projet |

|---|---|---|

| **B2.1** | Exploitation et administration des éléments d'une infrastructure | Installation et administration de Snort sur Ubuntu Server 22.04 |

| **B2.2** | Exploitation et administration d'un système de communication | Configuration de l'interface réseau en mode promiscuous, gestion des flux |

| **B2.3** | Automatisation des tâches d'administration système | Service systemd pour Snort, logrotate, scripts de monitoring |

| **B2.4** | Exploitation et administration d'un annuaire | (Win-server : AD optionnel) |

Bloc 3 – Cybersécurité des services informatiques

| Compétence | Intitulé | Mise en œuvre dans le projet |

|---|---|---|

| **B3.1** | Protéger les données à caractère personnel | Justification du projet par la protection des données de TechNova |

| **B3.2** | Préserver l'identité numérique de l'organisation | Détection des tentatives d'usurpation et d'intrusion |

| **B3.3** | Sécuriser les équipements et les usages des utilisateurs | Règles Snort anti-brute force, fail2ban |

| **B3.4** | Garantir la disponibilité, l'intégrité et la confidentialité des services | Supervision Zabbix du service Snort, mode IDS non intrusif |

| **B3.5** | Assurer la cybersécurité d'une solution applicative et de son développement | Tests d'attaques web (SQLi, directory traversal), règles HTTP |

🛠️ Compétences techniques détaillées

Administration Linux

- Installation depuis les sources (compilation Snort 3)
- Gestion des services systemd (`enable` , `start` , `status` , `restart`)
- Configuration réseau avancée (mode promiscuous, bridges)
- Gestion des logs et logrotate
- Droits et permissions (sudo, fichiers de configuration)

Réseaux

- Compréhension des protocoles TCP/IP (TCP flags, ICMP types, UDP)
- Analyse du trafic réseau (tcpdump, Wireshark)
- Segmentation réseau (VLAN, sous-réseaux)
- Notions de pare-feu et de filtrage de paquets

Sécurité informatique

- Principe de fonctionnement d'un IDS/IPS
- Écriture de règles de détection (syntaxe Snort)
- Reconnaissance des patterns d'attaque (scans, brute-force, DoS)
- Analyse forensique de logs réseau
- Connaissance des outils offensifs (Nmap, Hydra, hping3)

Virtualisation

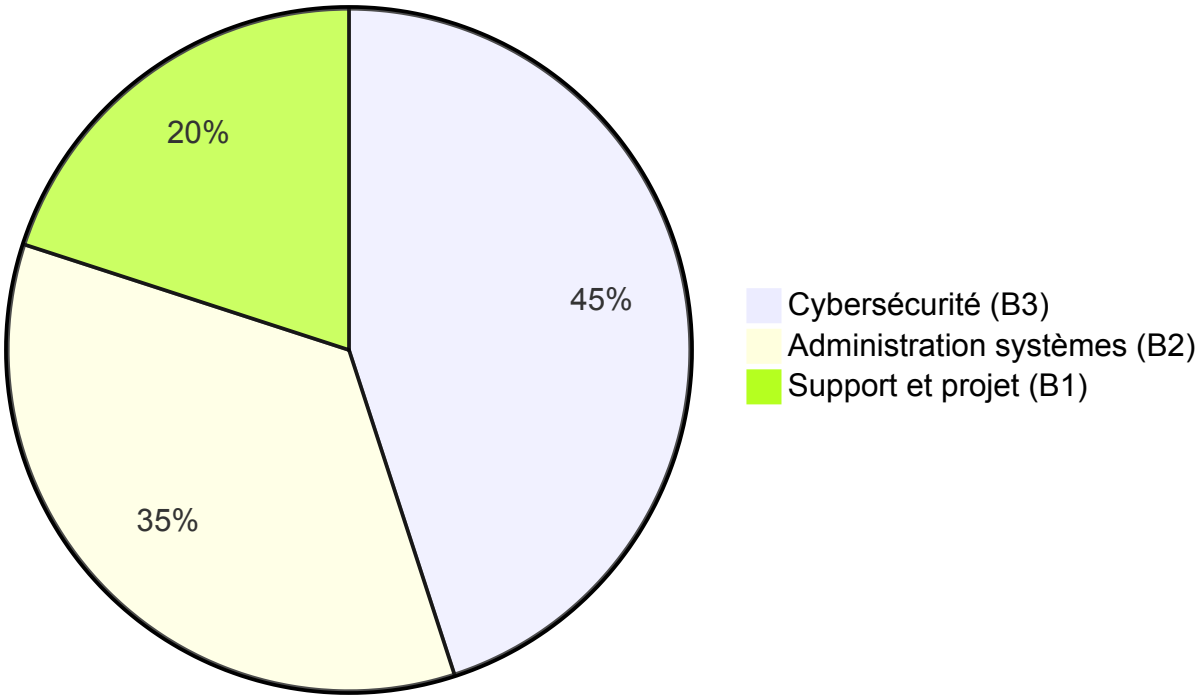
- Déploiement de VMs sous VMware Fusion / Proxmox VE
- Configuration des réseaux virtuels (host-only, internal)
- Gestion des ressources (vCPU, RAM, stockage)

Documentation

- Rédaction technique en Markdown (Obsidian)
- Schémas réseau (draw.io, Mermaid)
- Procédures d'installation step-by-step

📊 Tableau de couverture du référentiel E6

Répartition des compétences mobilisées



🏆 Compétences transversales

| Compétence | Description |

|---|---|

| **Autonomie** | Déploiement complet de l'infrastructure sans assistance externe |

| **Rigueur** | Documentation exhaustive de chaque étape |

| **Adaptabilité** | Résolution des problèmes rencontrés (compatibilité, erreurs de config) |

| **Communication** | Présentation orale du projet à un jury professionnel |

| **Veille technologique** | Utilisation de Snort 3, version la plus récente |

🔗 Navigation

← Retour : 11 - Sécurisation et améliorations

→ Suite : 13 - Conclusion

---

tags:

- BTS-SIO
- SISR
- E6
- Snort
- Conclusion

parents: "00 - Index du projet"

---

13 – Conclusion du projet

✅ Bilan des réalisations

Ce projet avait pour objectif la mise en place d'un système de détection d'intrusion réseau basé sur **Snort 3** dans une infrastructure entièrement virtualisée, simulant l'environnement

informatique de l'entreprise fictive **TechNova Solutions**.

Ce qui a été accompli

| Objectif | Statut |

|---|---|

| Déploiement de l'infrastructure virtualisée (5 VMs) | ✅ Réalisé |

| Installation et configuration de Snort 3 | ✅ Réalisé |

| Création de règles Snort personnalisées (15 règles) | ✅ Réalisé |

| Simulation d'attaques depuis Kali Linux | ✅ Réalisé |

| Détection des attaques par Snort | ✅ Réalisé |

| Analyse et interprétation des alertes | ✅ Réalisé |

| Documentation technique complète | ✅ Réalisé |

| Propositions d'améliorations | ✅ Réalisé |

📊 Résultats des tests

| Attaque simulée | Détectée par Snort | Temps de détection |

|---|---|---|

| Scan SYN (Nmap -sS) | ✅ Oui | < 1 seconde |

| Scan NULL (Nmap -sN) | ✅ Oui | < 1 seconde |

| Scan XMAS (Nmap -sX) | ✅ Oui | < 1 seconde |

| Scan UDP (Nmap -sU) | ✅ Oui | < 5 secondes |

| Brute-force SSH (Hydra) | ✅ Oui | < 3 secondes |

| Brute-force FTP (Hydra) | ✅ Oui | < 3 secondes |

| Ping flood (hping3) | ✅ Oui | < 1 seconde |

| Directory traversal | ✅ Oui | < 1 seconde |

| Scanner Nikto | ✅ Oui | < 2 secondes |

Taux de détection global : 100% (sur les attaques simulées)

🔍 Ce que ce projet démontre

Ce projet démontre qu'il est possible de mettre en place un IDS fonctionnel et efficace avec des **outils open-source gratuits**, tout en couvrant l'ensemble des aspects d'un projet d'infrastructure professionnel :

- **Planification** : analyse des besoins, choix des outils, définition de l'architecture
- **Déploiement** : installation et configuration pas-à-pas documentées
- **Tests** : simulation d'attaques réalistes et mesure des résultats
- **Exploitation** : lecture des alertes, interprétation, proposition d'actions correctives
- **Amélioration** : roadmap vers une solution encore plus complète (IPS, SIEM, Zabbix)

🚀 Perspectives d'évolution

Court terme (projet BTS)

- Intégrer les alertes Snort dans le projet Zabbix existant
- Passer en mode IPS inline pour bloquer automatiquement les attaques
- Déployer Wazuh pour la corrélation d'événements

Moyen terme (professionnel)

- Déployer Snort en production sur le réseau d'Elionis (avec accord)
- Mettre en place un SOC (Security Operations Center) léger
- Certifications : CompTIA Security+, CEH

💡 Apports personnels

Retour d'expérience

> Ce projet m'a permis d'acquérir une vision concrète du métier d'analyste en cybersécurité réseau. Comprendre comment un attaquant opère (avec Kali Linux) et comment un IDS réagit (avec Snort) donne une perspective précieuse sur les deux faces de la sécurité offensive et défensive. L'ensemble des compétences mobilisées — Linux, réseaux,

> virtualisation, documentation — s'inscrit directement dans mon parcours vers un **Bac+3 en Réseaux et Cybersécurité**.

📚 Ressources utilisées

| Ressource | Lien |

|---|---|

| Documentation officielle Snort 3 | https://docs.snort.org |

| Snort Rules Reference | https://www.snort.org/downloads |

| Kali Linux Tools | https://www.kali.org/tools |

| Référentiel BTS SIO | BOEN 2011 + CNAS |

| Portfolio personnel | https://mfiportfolio.fr |

🔗 Navigation

← Retour : 12 - Compétences BTS SIO SISR

← Index : 00 - Index du projet