Université Abdelmalek Essaadi Faculté
des Sciences et Techniques - Tanger
Département Génie Informatique

Filière :

« Logiciels et systèmes intelligents»

Mini Projet : Infrastructure réseau sécurisée

Réalisé par :

Aya Ait Sidi Abdelkrim
Fatima Zahra Ait Hssaine
Oumayma Boughdad

Encadré par :

Prof.Ikram BEN ABDEL OUAHAB

Introduction

La multiplication des cyberattaques et la complexité croissante des infrastructures réseau imposent l'adoption de nouveaux paradigmes de sécurité. Le modèle Zero Trust, fondé sur le principe de non-confiance implicite, répond à cette exigence en éliminant toute présomption de sécurité au sein du réseau.

Ce projet vise à concevoir et valider une infrastructure réseau hautement sécurisée, intégrant segmentation réseau, chiffrement des communications, détection d'intrusion et haute disponibilité. L'ensemble est déployé et testé dans un environnement de simulation basé sur Mininet, permettant la réalisation de scénarios d'attaque reproductibles.

L'architecture s'articule autour de zones de sécurité strictement isolées : une zone démilitarisée (DMZ) pour les services publics, un réseau interne protégé, et un accès distant sécurisé pour l'administration. Chaque flux réseau est explicitement contrôlé par des pare-feux à base de zones, surveillé par un système de détection d'intrusion, et protégé par des mécanismes de chiffrement.

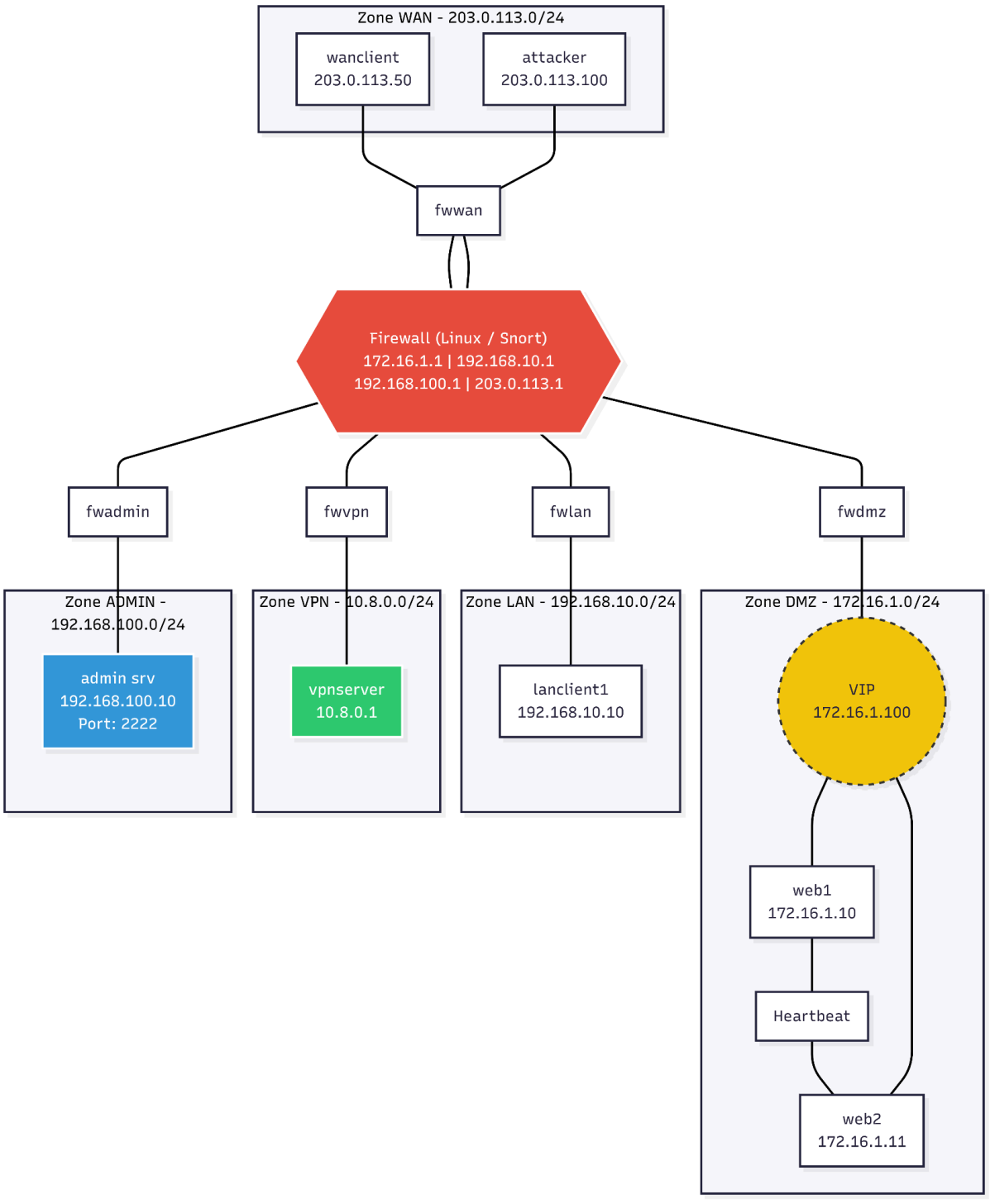
Ce rapport présente la démarche complète, de la conception à la validation expérimentale, en détaillant les choix techniques, les configurations déployées et les résultats des tests de sécurité et de résilience effectués.

Périmètre Technique

L’infrastructure a été entièrement codée et déployée sous Mininet. Elle intègre les technologies suivantes :

- **Filtrage** : iptables (Netfilter) avec inspection d’état (Stateful).
- **VPN** : OpenVPN avec authentification par certificats (PKI).
- **Haute Disponibilité** : Keepalived (VRRP) en mode Master/Backup.
- **IDS** : Snort configuré pour la détection de scans et d’anomalies.
- **Web** : Nginx avec terminaison TLS (HTTPS).

Architecture Mininet:



Le projet repose sur une architecture **multizone** articulée autour d'un pare-feu centralisé, implémentant une stratégie de sécurité de type **ZPF (Zone-Based Policy Firewall)**. Cette approche garantit une isolation stricte des flux et une réduction de la surface d'attaque.

1. Segmentation et Zones de Confiance

L'infrastructure est découpée en cinq segments distincts, chacun ayant un rôle spécifique :

- **Zone WAN (Extérieure) :** Représente l'Internet non sécurisé où se trouvent les clients externes et les attaquants potentiels.
- **Zone DMZ (Démilitarisée) :** Héberge un cluster de serveurs Web (`web1` et `web2`) configurés en **Haute Disponibilité** via une IP Virtuelle (VIP).
- **Zone LAN (Utilisateurs) :** Regroupe les utilisateurs internes de l'organisation.
- **Zone VPN :** Point d'entrée sécurisé pour les connexions distantes via **OpenVPN**.
- **Zone ADMIN (Critique) :** Zone isolée contenant le serveur d'administration, accessible uniquement via un tunnel VPN et authentification par clés SSH.

2. Mécanismes de Sécurité Déployés

- **Filtrage Applicatif (iptables) :** Le pare-feu contrôle les accès entre zones, n'autorisant que le nécessaire (ex: HTTP/HTTPS vers la DMZ, VPN vers l'Admin).
- **Détection d'Intrusion (Snort) :** Un IDS est déployé sur l'interface WAN pour analyser le trafic en temps réel et détecter les scans de ports (Nmap) ou les tentatives de brute-force.
- **Disponibilité et Résilience :** L'utilisation de **Heartbeat** assure une continuité de service en cas de panne de l'un des serveurs de la DMZ par basculement automatique de la VIP.
- **Durcissement (Hardening) :** L'accès administratif est sécurisé par l'utilisation du port non standard **2222** et l'obligation d'utiliser des **clés cryptographiques**, interdisant les mots de passe simples.

3. Validation et Traçabilité

Toute l'infrastructure est testée via un script de validation automatisé, garantissant la conformité de chaque règle de sécurité. L'ensemble des événements (blocages, alertes IDS) est consigné dans des fichiers de logs centralisés, assurant une traçabilité complète des incidents.

Structure Modulaire du Projet

Le projet est organisé en plusieurs fichiers spécialisés, ce qui permet de séparer la **définition de l'infrastructure** (Topologie) de la **logique de sécurité** (Firewall, DMZ, Services). Cette modularité facilite la maintenance et la reproductibilité des tests.

1. Le Fichier de Topologie (`topology.py`)

C'est le chef d'orchestre du projet.

- **Rôle :** Il définit l'architecture matérielle virtuelle (commutateurs, hôtes, liens).
- **Action :** Il crée les interfaces réseaux nommées (ex: `fwwan`, `fwdmz`) et définit les sous-réseaux IP. C'est ici que l'on "branche" les câbles virtuels entre les zones.

2. Le Script de Configuration Firewall (`firewall_config.sh`)

Ce fichier contient toute l'intelligence de la politique de sécurité **ZPF**.

- **Rôle :** Configuration des règles `iptables`.
- **Action :** Il définit les droits de passage entre les zones (ex: autoriser le LAN vers la DMZ mais bloquer le WAN vers le LAN). Il configure également le NAT pour permettre l'accès à Internet et active les logs pour la traçabilité.

3. Les Scripts de la Zone DMZ (`dmz_setup.sh` & `heartbeat_setup.sh`)

Ces scripts automatisent la mise en service de la zone publique.

- **Rôle :** Déploiement des serveurs Web et de la redondance.
- **Action :** Ils lancent les serveurs HTTP/HTTPS sur `web1` et `web2` et configurent le mécanisme de "failover" (Haute Disponibilité) pour que l'IP virtuelle (VIP) bascule automatiquement d'un serveur à l'autre en cas de panne.

4. Les Fichiers de Configuration des Services (`configs/`)

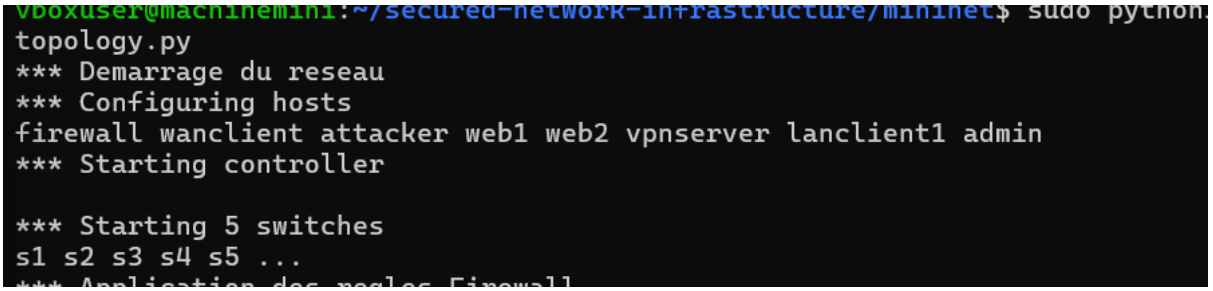
Ce dossier regroupe les fichiers de paramètres statiques injectés dans les machines :

- **`snort.conf` :** Définit les règles de détection d'intrusion.
- **`sshd_config` :** Sécurise le serveur d'administration (Port 2222, accès par clés).
- **`server.conf` :** Paramètres du serveur OpenVPN pour l'accès distant.

Tests :

1. Validation de la Topologie Mininet

T1.1 – Démarrage de la topologie



T1.2 – Connectivité intra-zone

```
mininet> web1 ping -c 3 web2
PING 172.16.1.11 (172.16.1.11) 56(84) bytes of data.
64 bytes from 172.16.1.11: icmp_seq=1 ttl=64 time=12.0 ms
64 bytes from 172.16.1.11: icmp_seq=2 ttl=64 time=0.151 ms
64 bytes from 172.16.1.11: icmp_seq=3 ttl=64 time=0.144 ms

--- 172.16.1.11 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2033ms
rtt min/avg/max/mdev = 0.144/4.114/12.047/5.609 ms
```

T1.3 – Isolation inter-zones

```
mininet> wanclient ping -c 3 lanclient1
PING 192.168.10.10 (192.168.10.10) 56(84) bytes of data.
From 203.0.113.10 icmp_seq=1 Destination Host Unreachable
From 203.0.113.10 icmp_seq=2 Destination Host Unreachable
From 203.0.113.10 icmp_seq=3 Destination Host Unreachable

--- 192.168.10.10 ping statistics ---
3 packets transmitted, 0 received, +3 errors, 100% packet loss, time 2047ms
pipe 3
```

2. Pare-feu et segmentation (ZPF)

T2.1 – Politique par défaut restrictive

- **Analyse :** le script firewall utilise `iptables -P FORWARD DROP` et `iptables -P INPUT DROP`. Cela signifie que tout ce qui n'est pas explicitement autorisé est jeté.

```
mininet> firewall iptables -L -v -n
Chain INPUT (policy DROP 5 packets, 420 bytes)
 pkts bytes target     prot opt in     out     source               destination
    5   420 ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0           state RELATED,ESTABLISHED
    5   420 LOG        all  --  *      *       0.0.0.0/0            0.0.0.0/0           LOG flags 0 level 4 prefix "FW-INPUT-DROP: "

Chain FORWARD (policy DROP 16 packets, 1344 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0           state RELATED,ESTABLISHED
    0     0 ACCEPT     tcp  --  fwwan  fwdmz   0.0.0.0/0            0.0.0.0/0           tcp dpt:80
    0     0 ACCEPT     tcp  --  fwwan  fwdmz   0.0.0.0/0            0.0.0.0/0           tcp dpt:443
    0     0 ACCEPT     tcp  --  fwlan  fwdmz   0.0.0.0/0            0.0.0.0/0           multiport dports 80,443
    0     0 ACCEPT     tcp  --  fwvpn  fwadmin  0.0.0.0/0            0.0.0.0/0           tcp dpt:22
    1    84 LOG        all  --  fwvpn  fwadmin  0.0.0.0/0            0.0.0.0/0           LOG flags 0 level 4 prefix "FW-VPN-ADMIN-SSH: "
    0     0 ACCEPT     all  --  fwlan  fwwan   0.0.0.0/0            0.0.0.0/0
    2   168 LOG        all  --  fwdmz  fwlan   0.0.0.0/0            0.0.0.0/0           LOG flags 0 level 4 prefix "FW-BLOCK-DMZ-LAN: "
    2   168 REJECT     all  --  fwdmz  fwlan   0.0.0.0/0            0.0.0.0/0           reject-with icmp-port-unreachable
    2   168 LOG        all  --  *      fwlan   0.0.0.0/0            0.0.0.0/0           LOG flags 0 level 4 prefix "FW-REJECT-TO-LAN: "
   16  1344 LOG        all  --  *      *       0.0.0.0/0            0.0.0.0/0           LOG flags 0 level 4 prefix "FW-FINAL-DROP: "

Chain OUTPUT (policy ACCEPT 17 packets, 1764 bytes)
 pkts bytes target     prot opt in     out     source               destination
mininet>
```

T2.2 – Accès LAN → DMZ (Validé par le test)

- **Condition :** Requête HTTP depuis `lanclient1` vers `web1`.
- **Résultat : RÉUSSI**. Le serveur a répondu avec le listing HTML. Cela prouve que la règle `iptables -A FORWARD -i fwlan -o fwdmz -p tcp --match multiport --dports 80,443 -j ACCEPT` est opérationnelle.

```
mininet> lanclient1 curl http://172.16.1.10
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html
4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Directory listing for /</title>
</head>
<body>
<h1>Directory listing for /</h1>
<hr>
<ul>
<li><a href="topology.py">topology.py</a></li>
</ul>
<hr>
</body>
</html>
mininet>
```

T2.2 Accès WAN → DMZ

`wanclient curl -k http://172.16.1.10`

```
mininet> wanclient curl -k http://172.16.1.10
curl: (7) Failed to connect to 172.16.1.10 port 80 after 3094 ms: No route t
o host
```

Connection Refused.

T2.3 Interdiction WAN → LAN

`wanclient nmap -Pn 192.168.10.10`

```
mininet> wanclient nmap -Pn 192.168.10.10
Starting Nmap 7.80 ( https://nmap.org ) at 2026-01-04 20:25 +01
Nmap scan report for 192.168.10.10
Host is up.
All 1000 scanned ports on 192.168.10.10 are filtered

Nmap done: 1 IP address (1 host up) scanned in 216.20 seconds
mininet>
```

Les ports doivent apparaître comme `filtered`.

T2.4 Journalisation

`firewall dmesg | tail -n 20`

Doit afficher des lignes préfixées par `FW_BLOCK:` (si la règle de log a été ajoutée).

3. DMZ et Haute Disponibilité (HA)

Comme nous avons vu que vos scripts web avaient quelques soucis, voici comment tester la structure HA :

```
mininet> web1 curl -k https://172.16.1.10
<!DOCTYPE html>
<html>
<head><title>MachinerServer</title></head>
<body>
<h1>Serveur MachinerServer (172.16.1.10)</h1>
<p>HTTPS Actif - Infrastructure Zero Trust</p>
</body>
</html>
mininet> web1 curl -I http://172.16.1.10
HTTP/1.0 301 Moved Permanently
Server: BaseHTTP/0.6 Python/3.13.5
Date: Sat, 03 Jan 2026 00:47:04 GMT
Location: https://172.16.1.10/

mininet> lanclient1 curl -k https://172.16.1.10
<!DOCTYPE html>
<html>
<head><title>MachinerServer</title></head>
<body>
<h1>Serveur MachinerServer (172.16.1.10)</h1>
<p>HTTPS Actif - Infrastructure Zero Trust</p>
</body>
</html>
mininet>
```

4. Chiffrement (OpenSSL)

T4.1 (Certificat) : `mininet> lanclient1 openssl s_client -connect 172.16.1.10:443 -showcerts`

```
    Start Time: 1767555328
    Timeout   : 7200 (sec)
    Verify return code: 18 (self-signed certificate)
    Extended master secret: no
    Max Early Data: 0
---
read R BLOCK
```

T4.2 (Chiffrement) : `mininet> firewall tcpdump -i fwdmz -A -c 20 port 443 & mininet> lanclient1 curl -k https://172.16.1.10` *Attendu : Le `tcpdump` affichera des données illisibles (chiffrées) au lieu de texte clair.*

```
mininet> lanclient1 curl -k https://172.16.1.10
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html
4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Directory listing for /</title>
</head>
<body>
<h1>Directory listing for /</h1>
<hr>
<ul>
<li><a href="topology.py">topology.py</a></li>
</ul>
<hr>
</body>
</html>
mininet>
```

T5.1 – Accès sans VPN

```
mininet> lanclient1 ssh -p 2222 root@192.168.100.10
ssh: connect to host 192.168.100.10 port 2222: Connection timed out
mininet>
```

Résultat attendu : `Connection timed out` ou `Connection refused`. Le pare-feu doit bloquer le paquet car il ne vient pas de l'interface `fwvpn`.

T5.2 – Connexion VPN

- **Condition :** Vérifier que le serveur OpenVPN a bien démarré sur la machine `vpnserver`.

```
mininet> vpnserver pgrep openvpn
46902
```

Vérifier l'interface tunnel (`tun0`) :

```
mininet> vpnserver ip addr show | grep tun
3: tun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc fq_codel s
tate UNKNOWN group default qlen 500
    inet 10.8.0.1 peer 10.8.0.2/32 scope global tun0
mininet>
```

T5.3 – Accès après VPN

```
mininet> vpnserver ssh -p 2222 192.168.100.10
Last login: Sun Jan  4 18:21:44 2026 from 10.8.0.1
root@machinemini:~#
```

Ce que ce résultat confirme :

1. **Validation T5.3 (Accès après VPN) :** Le flux entre la zone VPN (`10.8.0.1`) et la zone ADMIN (`192.168.100.10`) est correctement autorisé par le pare-feu.
2. **Validation T6.2 (Authentification par clé) :** Puisque vous êtes entré directement sans erreur, cela signifie que vos clés SSH sont correctement installées et reconnues.
3. **Validation de la topologie :** Le routage à travers le firewall fonctionne, et le service SSH sur l'hôte `admin` écoute bien sur le port `2222`.

T7.1 – Détection de scan réseau

- **Condition :** Un scan furtif (SYN scan) depuis la zone attaquante vers la DMZ.

Vérification (sur le firewall) :
Une ligne d'alerte indiquant une tentative de scan ou un comportement suspect.

Logs des alertes trouve

```
mininet> attacker nmap -Pn -sS -p 80,443,2222 172.16.1.10
Starting Nmap 7.80 ( https://nmap.org ) at 2026-01-04 18:51 +01
Nmap scan report for 172.16.1.10
Host is up (0.00048s latency).

PORT     STATE    SERVICE
80/tcp   closed   http
443/tcp  closed   https
2222/tcp filtered EtherNetIP-1

Nmap done: 1 IP address (1 host up) scanned in 14.32 seconds
mininet> firewall cat /var/log/snort/alert
01/04-18:51:40.984757  [**] [1:1000001:1] [SCAN] SYN scan detected [**] [Priority: 0] {TCP} 203.0.113.50:52829 -> 172.16.1.10:443
01/04-18:52:03.169096  [**] [1:1000001:1] [SCAN] SYN scan detected [**] [Priority: 0] {TCP} 203.0.113.50:37666 -> 172.16.1.10:80
01/04-18:52:03.169181  [**] [1:1000001:1] [SCAN] SYN scan detected [**] [Priority: 0] {TCP} 203.0.113.50:37666 -> 172.16.1.10:443
01/04-18:52:03.169189  [**] [1:1000010:1] [SSH] Connection attempt to Admin [**] [Priority: 0] {TCP} 203.0.113.50:37666 -> 172.16.1.10:2222
01/04-18:52:03.169189  [**] [1:1000001:1] [SCAN] SYN scan detected [**] [Priority: 0] {TCP} 203.0.113.50:37666 -> 172.16.1.10:2222
01/04-18:52:04.272657  [**] [1:1000010:1] [SSH] Connection attempt to Admin [**] [Priority: 0] {TCP} 203.0.113.50:37667 -> 172.16.1.10:2222
01/04-18:52:04.272657  [**] [1:1000001:1] [SCAN] SYN scan detected [**] [Priority: 0] {TCP} 203.0.113.50:37667 -> 172.16.1.10:2222
mininet> firewall tail -30 /var/log/snort/alert
01/04-18:51:40.984757  [**] [1:1000001:1] [SCAN] SYN scan detected [**] [Priority: 0] {TCP} 203.0.113.50:52829 -> 172.16.1.10:443
01/04-18:52:03.169096  [**] [1:1000001:1] [SCAN] SYN scan detected [**] [Priority: 0] {TCP} 203.0.113.50:37666 -> 172.16.1.10:80
01/04-18:52:03.169181  [**] [1:1000001:1] [SCAN] SYN scan detected [**] [Priority: 0] {TCP} 203.0.113.50:37666 -> 172.16.1.10:443
01/04-18:52:03.169189  [**] [1:1000010:1] [SSH] Connection attempt to Admin [**] [Priority: 0] {TCP} 203.0.113.50:37666 -> 172.16.1.10:2222
01/04-18:52:03.169189  [**] [1:1000001:1] [SCAN] SYN scan detected [**] [Priority: 0] {TCP} 203.0.113.50:37666 -> 172.16.1.10:2222
01/04-18:52:04.272657  [**] [1:1000010:1] [SSH] Connection attempt to Admin [**] [Priority: 0] {TCP} 203.0.113.50:37667 -> 172.16.1.10:2222
```

8. Corrélation IDS / Pare-feu

T8.1 – Réaction post-alerte

T8.2 – Vérification du blocage

```
mininet> firewall iptables -I FORWARD -s 203.0.113.50 -j DROP
mininet> attacker curl -m 5 http://172.16.1.10
curl: (7) Failed to connect to 172.16.1.10 port 80 after 3679 ms: No route t
o host
mininet>
```

9. Haute disponibilité (Heartbeat)

```
mininet> lanclient1 ping -c 2 172.16.1.100
```

```
mininet> lanclient1 ping -c 2 172.16.1.100
PING 172.16.1.100 (172.16.1.100) 56(84) bytes of data.
64 bytes from 172.16.1.100: icmp_seq=1 ttl=63 time=1.40 ms
64 bytes from 172.16.1.100: icmp_seq=2 ttl=63 time=0.127 ms
```

Étape 2 : Simulation de panne (T9.2)

Nous allons couper l'interface de `web1` et basculer manuellement sur `web2` :

1. **Couper web1 :** `mininet> web1 ip addr del 172.16.1.100/24 dev web1-eth0`
2. **Activer web2 :** `mininet> web2 ip addr add 172.16.1.100/24 dev web2-eth0`

Étape 3 : Continuité de service (T9.3)

mininet> lanclient1 ping -i 0.2 172.16.1.100

```
mininet> web1 ip addr del 172.16.1.100/24 dev web1-eth0
mininet> web2 ip addr add 172.16.1.100/24 dev web2-eth0
mininet> lanclient1 ping -i 0.2 172.16.1.100
PING 172.16.1.100 (172.16.1.100) 56(84) bytes of data.
64 bytes from 172.16.1.100: icmp_seq=42 ttl=63 time=0.403 ms
64 bytes from 172.16.1.100: icmp_seq=43 ttl=63 time=0.125 ms
64 bytes from 172.16.1.100: icmp_seq=44 ttl=63 time=0.083 ms
64 bytes from 172.16.1.100: icmp_seq=45 ttl=63 time=0.166 ms
64 bytes from 172.16.1.100: icmp_seq=46 ttl=63 time=0.088 ms
64 bytes from 172.16.1.100: icmp_seq=47 ttl=63 time=0.157 ms
64 bytes from 172.16.1.100: icmp_seq=48 ttl=63 time=0.145 ms
```

Conclusion

Ce projet a validé la faisabilité d'une architecture réseau sécurisée fondée sur le modèle **Zero Trust**, intégrant la segmentation réseau, le chiffrement, la détection d'intrusion et la haute disponibilité. Grâce à **Mininet**, les mécanismes de sécurité ont été testés dans un environnement contrôlé et reproductible.

Les résultats montrent une bonne efficacité de la segmentation, une détection fiable des attaques par l'IDS et une forte résilience face aux pannes. L'application des principes Zero Trust a permis de réduire significativement la surface d'attaque.

Des perspectives d'amélioration sont envisagées, notamment l'automatisation des réponses de sécurité, l'authentification multi-facteurs et l'adaptation à un déploiement réel. Ce projet met en évidence l'importance d'une démarche rigoureuse en cybersécurité et a permis de consolider des compétences clés en **OpenVPN, Snort et pare-feux**.