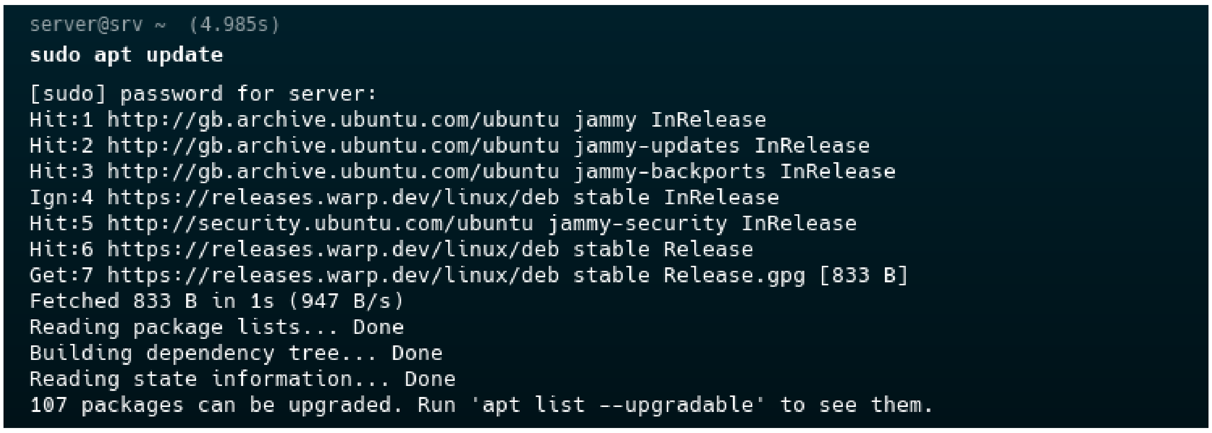
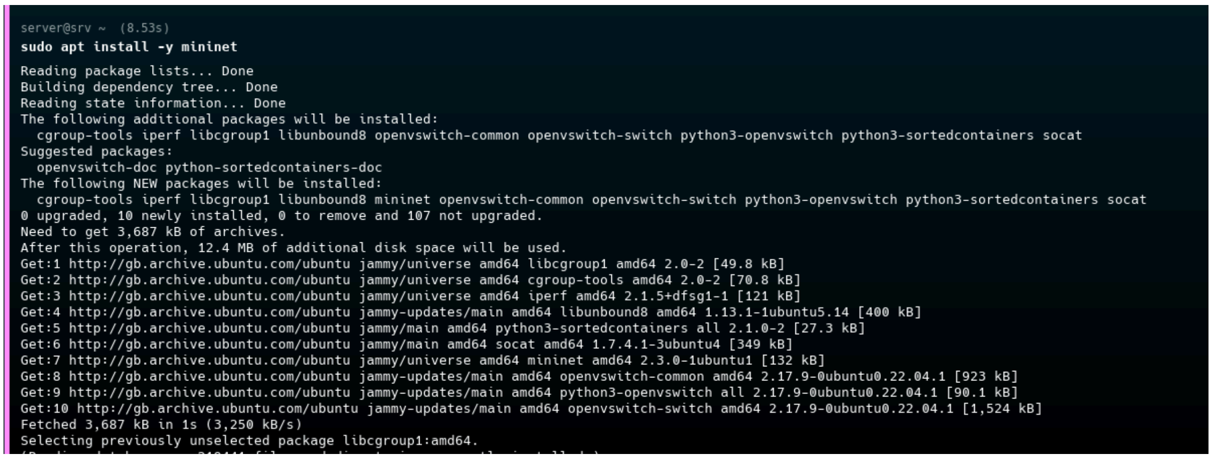
Étape 2 : Préparation de l'environnement

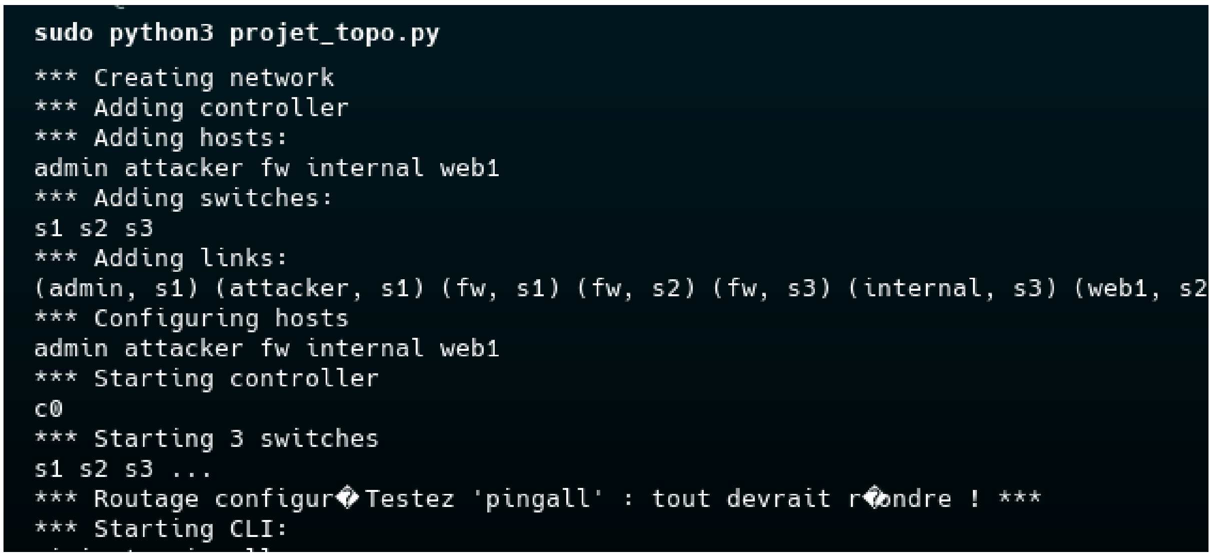


```
server@srv ~  (4.985s)
sudo apt update

[sudo] password for server:
Hit:1 http://gb.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://gb.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://gb.archive.ubuntu.com/ubuntu jammy-backports InRelease
Ign:4 https://releases.warp.dev/linux/deb stable InRelease
Hit:5 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:6 https://releases.warp.dev/linux/deb stable Release
Get:7 https://releases.warp.dev/linux/deb stable Release.gpg [833 B]
Fetched 833 B in 1s (947 B/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
107 packages can be upgraded. Run 'apt list --upgradable' to see them.
```



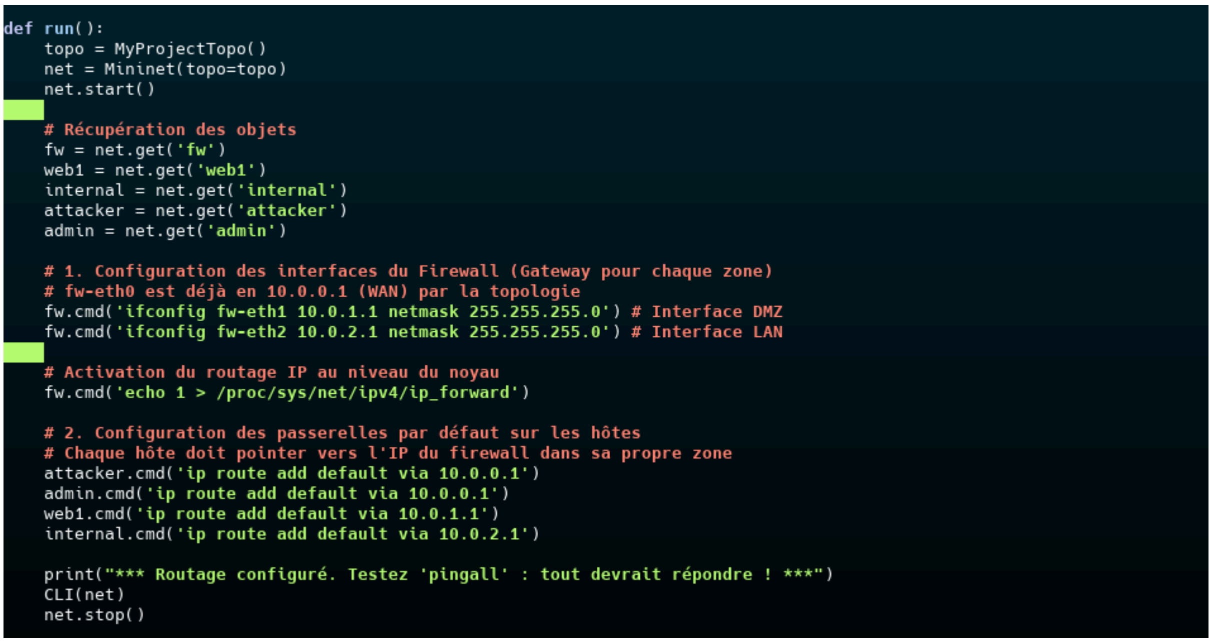
```
server@srv ~  (8.53s)
sudo apt install -y mininet

Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  cgroup-tools iperf libcgroup1 libunbound8 openvswitch-common openvswitch-switch python3-openvswitch python3-sortedcontainers socat
Suggested packages:
  openvswitch-doc python-sortedcontainers-doc
The following NEW packages will be installed:
  cgroup-tools iperf libcgroup1 libunbound8 mininet openvswitch-common openvswitch-switch python3-openvswitch python3-sortedcontainers socat
0 upgraded, 10 newly installed, 0 to remove and 107 not upgraded.
Need to get 3,687 kB of archives.
After this operation, 12.4 MB of additional disk space will be used.
Get:1 http://gb.archive.ubuntu.com/ubuntu jammy/universe amd64 libcgroup1 amd64 2.0-2 [49.8 kB]
Get:2 http://gb.archive.ubuntu.com/ubuntu jammy/universe amd64 cgroup-tools amd64 2.0-2 [70.8 kB]
Get:3 http://gb.archive.ubuntu.com/ubuntu jammy/universe amd64 iperf amd64 2.1.5+dfsg1-1 [121 kB]
Get:4 http://gb.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libunbound8 amd64 1.13.1-1ubuntu5.14 [400 kB]
Get:5 http://gb.archive.ubuntu.com/ubuntu jammy/main amd64 python3-sortedcontainers all 2.1.0-2 [27.3 kB]
Get:6 http://gb.archive.ubuntu.com/ubuntu jammy/main amd64 socat amd64 1.7.4.1-3ubuntu4 [349 kB]
Get:7 http://gb.archive.ubuntu.com/ubuntu jammy/universe amd64 mininet amd64 2.3.0-1ubuntu1 [132 kB]
Get:8 http://gb.archive.ubuntu.com/ubuntu jammy-updates/main amd64 openvswitch-common amd64 2.17.9-0ubuntu0.22.04.1 [923 kB]
Get:9 http://gb.archive.ubuntu.com/ubuntu jammy-updates/main amd64 python3-openvswitch all 2.17.9-0ubuntu0.22.04.1 [90.1 kB]
Get:10 http://gb.archive.ubuntu.com/ubuntu jammy-updates/main amd64 openvswitch-switch amd64 2.17.9-0ubuntu0.22.04.1 [1,524 kB]
Fetched 3,687 kB in 1s (3,250 kB/s)
Selecting previously unselected package libcgroup1:amd64.
```

Étape 3 : Création de la Topologie Mininet (Script Python)

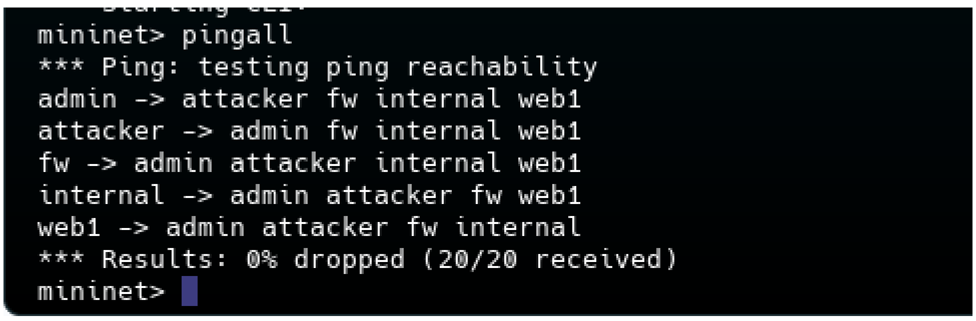


```
sudo python3 projet_topo.py

*** Creating network
*** Adding controller
*** Adding hosts:
admin attacker fw internal web1
*** Adding switches:
s1 s2 s3
*** Adding links:
(admin, s1) (attacker, s1) (fw, s1) (fw, s2) (fw, s3) (internal, s3) (web1, s2
*** Configuring hosts
admin attacker fw internal web1
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...
*** Routage configur�Testez 'pingall' : tout devrait r�ondre ! ***
*** Starting CLI:
```

Étape 4 : Configuration du Routage (La "tuyauterie")

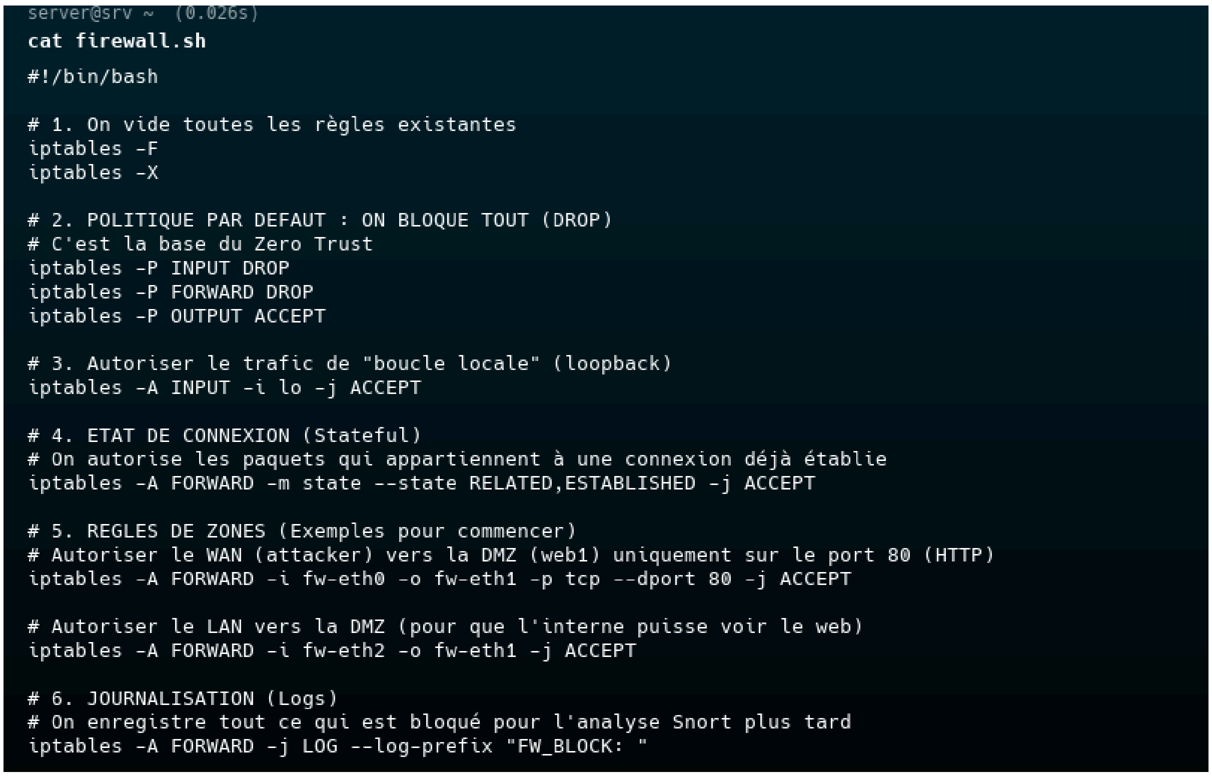


```
def run():
    topo = MyProjectTopo()
    net = Mininet(topo=topo)
    net.start()

    # Récupération des objets
    fw = net.get('fw')
    web1 = net.get('web1')
    internal = net.get('internal')
    attacker = net.get('attacker')
    admin = net.get('admin')

    # 1. Configuration des interfaces du Firewall (Gateway pour chaque zone)
    # fw-eth0 est déjà en 10.0.0.1 (WAN) par la topologie
    fw.cmd('ifconfig fw-eth1 10.0.1.1 netmask 255.255.255.0') # Interface DMZ
    fw.cmd('ifconfig fw-eth2 10.0.2.1 netmask 255.255.255.0') # Interface LAN

    # Activation du routage IP au niveau du noyau
    fw.cmd('echo 1 > /proc/sys/net/ipv4/ip_forward')

    # 2. Configuration des passerelles par défaut sur les hôtes
    # Chaque hôte doit pointer vers l'IP du firewall dans sa propre zone
    attacker.cmd('ip route add default via 10.0.0.1')
    admin.cmd('ip route add default via 10.0.0.1')
    web1.cmd('ip route add default via 10.0.1.1')
    internal.cmd('ip route add default via 10.0.2.1')

    print("*** Routage configuré. Testez 'pingall' : tout devrait répondre ! ***")
    CLI(net)
    net.stop()
```



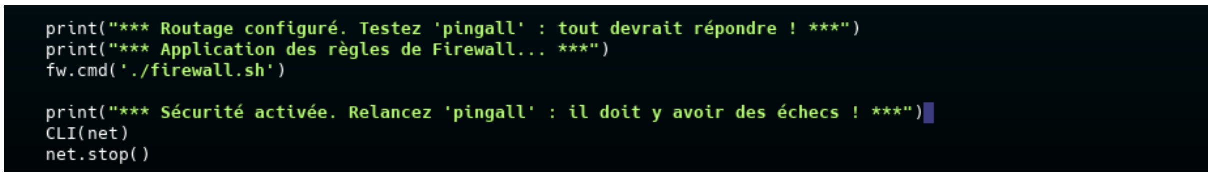
```
mininet> pingall
*** Ping: testing ping reachability
admin -> attacker fw internal web1
attacker -> admin fw internal web1
fw -> admin attacker internal web1
internal -> admin attacker fw web1
web1 -> admin attacker fw internal
*** Results: 0% dropped (20/20 received)
mininet>
```

Étape 5 : Mise en place du Pare-feu (Zone-Based Policy):



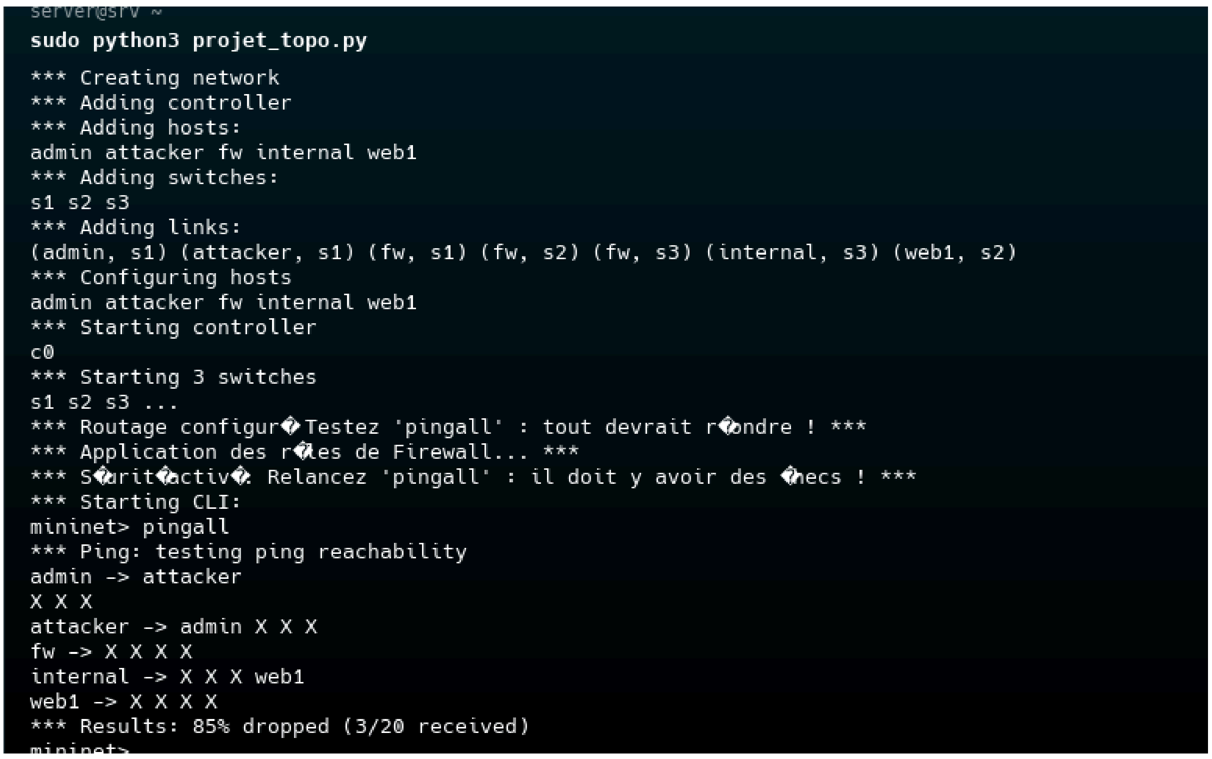
```
server@srv ~  (0.026s)
cat firewall.sh
#!/bin/bash

# 1. On vide toutes les règles existantes
iptables -F
iptables -X

# 2. POLITIQUE PAR DEFAUT : ON BLOQUE TOUT (DROP)
# C'est la base du Zero Trust
iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -P OUTPUT ACCEPT

# 3. Autoriser le trafic de "boucle locale" (loopback)
iptables -A INPUT -i lo -j ACCEPT

# 4. ETAT DE CONNEXION (Stateful)
# On autorise les paquets qui appartiennent à une connexion déjà établie
iptables -A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT

# 5. REGLES DE ZONES (Exemples pour commencer)
# Autoriser le WAN (attacker) vers la DMZ (web1) uniquement sur le port 80 (HTTP)
iptables -A FORWARD -i fw-eth0 -o fw-eth1 -p tcp --dport 80 -j ACCEPT

# Autoriser le LAN vers la DMZ (pour que l'interne puisse voir le web)
iptables -A FORWARD -i fw-eth2 -o fw-eth1 -j ACCEPT

# 6. JOURNALISATION (Logs)
# On enregistre tout ce qui est bloqué pour l'analyse Snort plus tard
iptables -A FORWARD -j LOG --log-prefix "FW_BLOCK: "
```

2. Intégration dans Mininet



```
    print("*** Routage configuré. Testez 'pingall' : tout devrait répondre ! ***")
    print("*** Application des règles de Firewall... ***")
    fw.cmd('./firewall.sh')

    print("*** Sécurité activée. Relancez 'pingall' : il doit y avoir des échecs ! ***")
    CLI(net)
    net.stop()
```

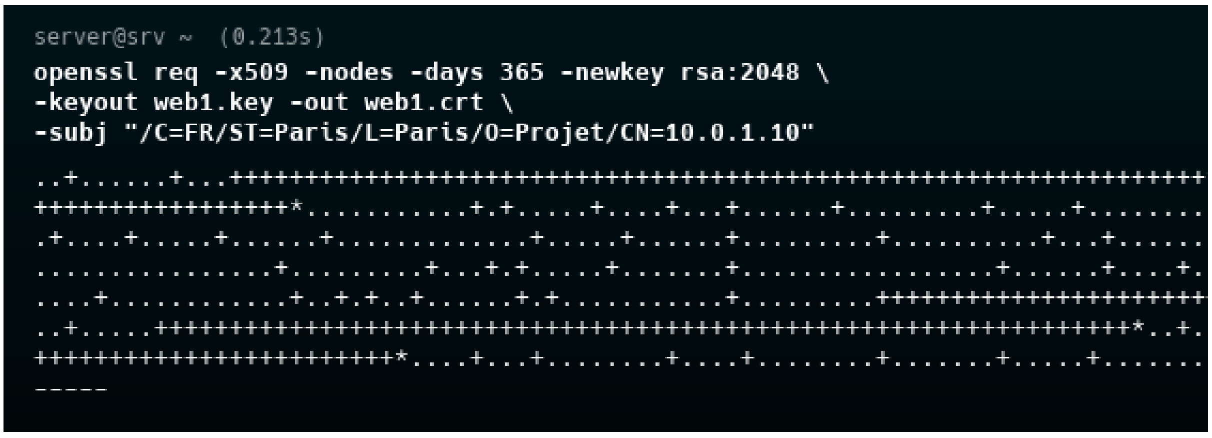
Étape 6 : Test de Sécurité (Le "Crash Test")



```
server@srv ~
sudo python3 projet_topo.py
*** Creating network
*** Adding controller
*** Adding hosts:
admin attacker fw internal web1
*** Adding switches:
s1 s2 s3
*** Adding links:
(admin, s1) (attacker, s1) (fw, s1) (fw, s2) (fw, s3) (internal, s3) (web1, s2)
*** Configuring hosts
admin attacker fw internal web1
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...
*** Routage configur� Testez 'pingall' : tout devrait r�ondre ! ***
*** Application des r�les de Firewall... ***
*** S�urit�activ�. Relancez 'pingall' : il doit y avoir des �hecs ! ***
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
admin -> attacker
X X X
attacker -> admin X X X
fw -> X X X X
internal -> X X X web1
web1 -> X X X X
*** Results: 85% dropped (3/20 received)
mininet>
```

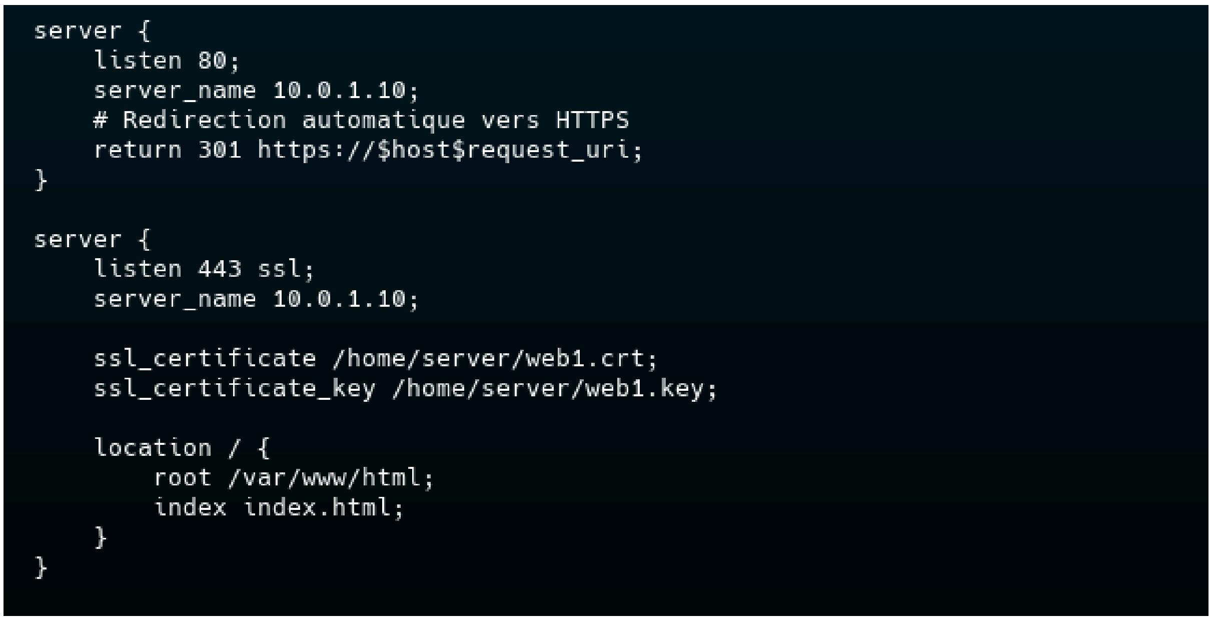
Étape 6 : Sécurisation HTTPS et Certificats (Point 5.2 du projet)

1. Génération du certificat SSL (OpenSSL)



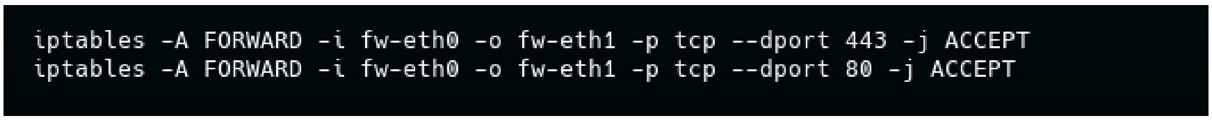
```
server@srv ~  (0.213s)
openssl req -x509 -nodes -days 365 -newkey rsa:2048 \
-keyout web1.key -out web1.crt \
-subj "/C=FR/ST=Paris/L=Paris/O=Projet/CN=10.0.1.10"

..+......+...+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++++++++++++*..........+.+.....+....+...+......+.........+.....+........
.+....+.....+......+............+.....+......+.........+.........+...+......
...............+.........+...+.+.....+.......+................+......+....+.
....+...........+..+.+..+......+.+..........+.........+++++++++++++++++++++++
..+.....+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*..+.
+++++++++++++++++++++++++*....+...+........+....+.........+.......+.....+.......
-----
```

2. Configuration du serveur Web avec redirection(fichier web1_conf):



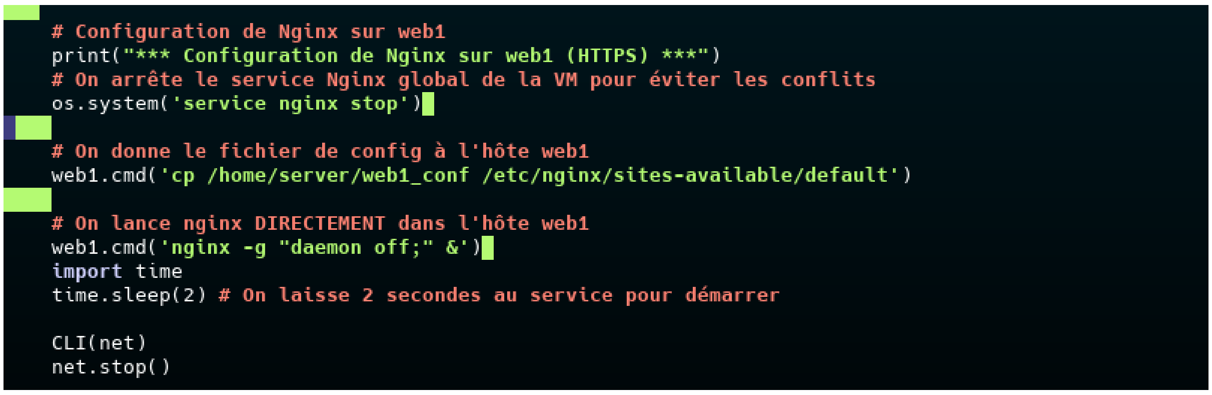
```
server {
    listen 80;
    server_name 10.0.1.10;
    # Redirection automatique vers HTTPS
    return 301 https://$host$request_uri;
}

server {
    listen 443 ssl;
    server_name 10.0.1.10;

    ssl_certificate /home/server/web1.crt;
    ssl_certificate_key /home/server/web1.key;

    location / {
        root /var/www/html;
        index index.html;
    }
}
```

3. Mise à jour du Pare-feu



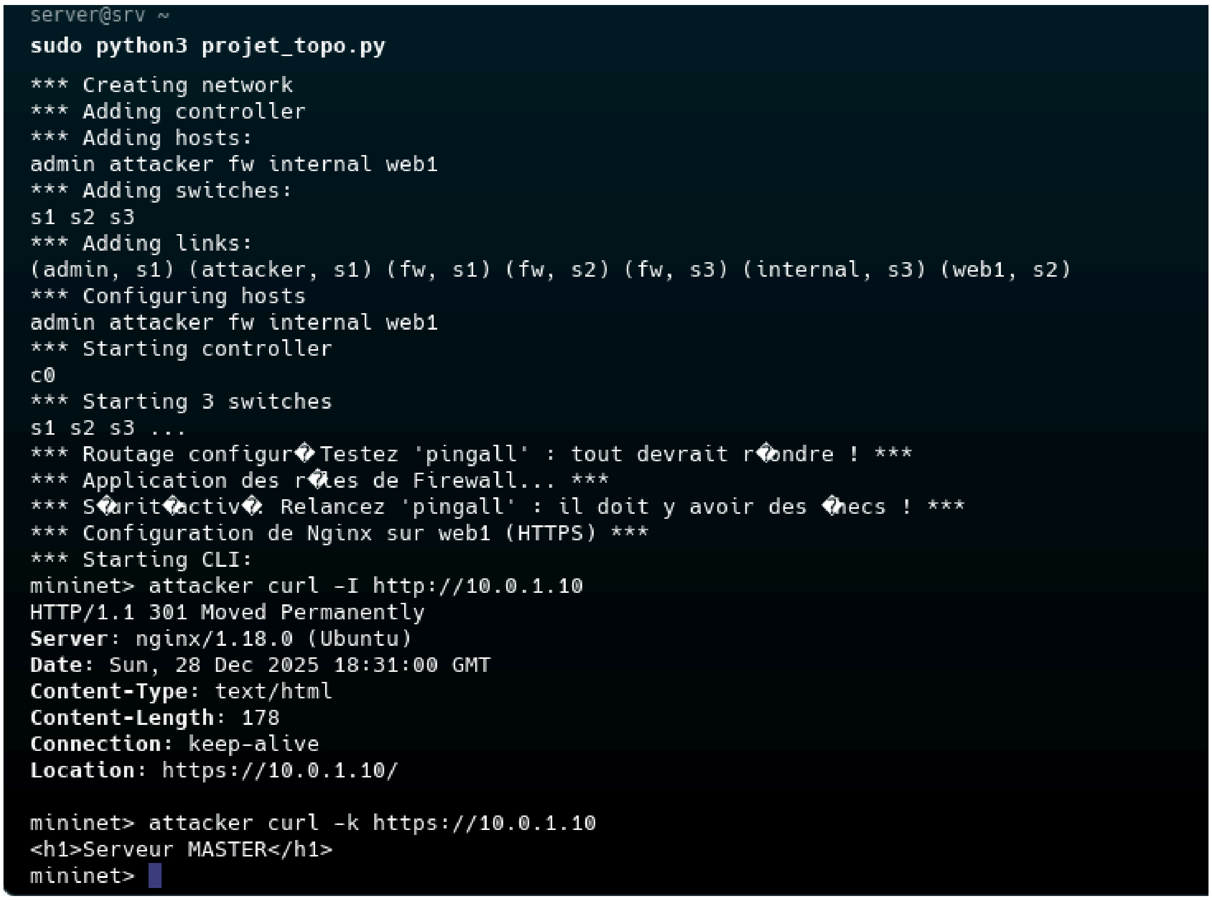
```
iptables -A FORWARD -i fw-eth0 -o fw-eth1 -p tcp --dport 443 -j ACCEPT
iptables -A FORWARD -i fw-eth0 -o fw-eth1 -p tcp --dport 80 -j ACCEPT
```

4.Automatiser le lancement dans ton script Python



```python
# Configuration de Nginx sur web1
print("*** Configuration de Nginx sur web1 (HTTPS) ***")
# On arrête le service Nginx global de la VM pour éviter les conflits
os.system('service nginx stop')

# On donne le fichier de config à l'hôte web1
web1.cmd('cp /home/server/web1_conf /etc/nginx/sites-available/default')

# On lance nginx DIRECTEMENT dans l'hôte web1
web1.cmd('nginx -g "daemon off;" &')
import time
time.sleep(2) # On laisse 2 secondes au service pour démarrer

CLI(net)
net.stop()
```

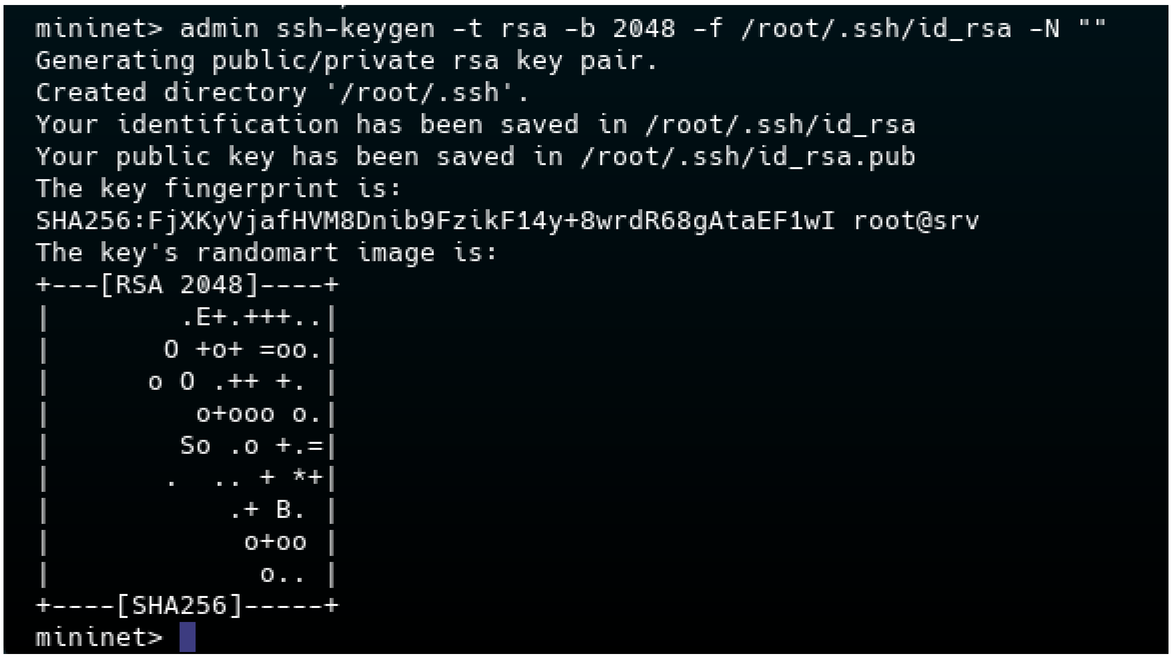
Verification:



```
server@srv ~
sudo python3 projet_topo.py
*** Creating network
*** Adding controller
*** Adding hosts:
admin attacker fw internal web1
*** Adding switches:
s1 s2 s3
*** Adding links:
(admin, s1) (attacker, s1) (fw, s1) (fw, s2) (fw, s3) (internal, s3) (web1, s2)
*** Configuring hosts
admin attacker fw internal web1
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...
*** Routage configur� Testez 'pingall' : tout devrait r�ondre ! ***
*** Application des r�les de Firewall... ***
*** S�urit�activ�: Relancez 'pingall' : il doit y avoir des �hecs ! ***
*** Configuration de Nginx sur web1 (HTTPS) ***
*** Starting CLI:
mininet> attacker curl -I http://10.0.1.10
HTTP/1.1 301 Moved Permanently
Server: nginx/1.18.0 (Ubuntu)
Date: Sun, 28 Dec 2025 18:31:00 GMT
Content-Type: text/html
Content-Length: 178
Connection: keep-alive
Location: https://10.0.1.10/

mininet> attacker curl -k https://10.0.1.10
<h1>Serveur MASTER</h1>
mininet>
```

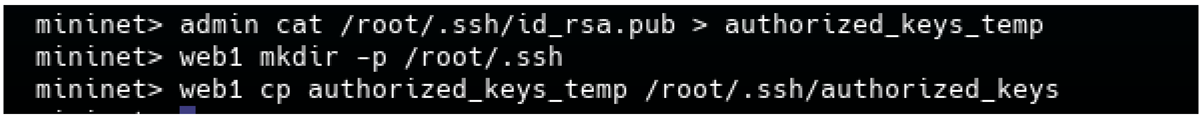
Étape 8 : Administration sécurisée (SSH par clé)

1. Générer les clés sur l'hôte `admin`



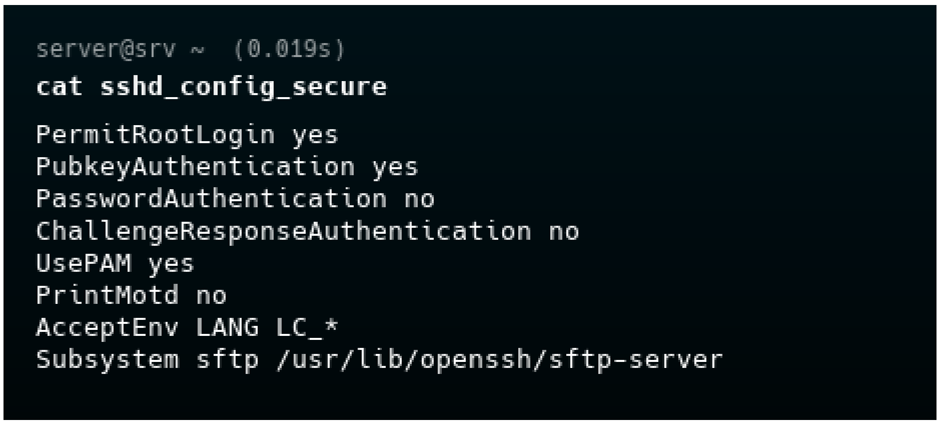
```
mininet> admin ssh-keygen -t rsa -b 2048 -f /root/.ssh/id_rsa -N ""
Generating public/private rsa key pair.
Created directory '/root/.ssh'.
Your identification has been saved in /root/.ssh/id_rsa
Your public key has been saved in /root/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:FjXKyVjafHVM8Dnib9FzikF14y+8wrdR68gAtaEF1wI root@srv
The key's randomart image is:
+---[RSA 2048]----+
|        .E+.+++..|
|       O +o+ =oo.|
|      o O .++ +. |
|         o+ooo o.|
|        So .o +.=|
|       .  .. + *+|
|           .+ B. |
|            o+oo |
|             o.. |
+----[SHA256]-----+
mininet>
```

2. "Déployer" la clé sur le serveur web1



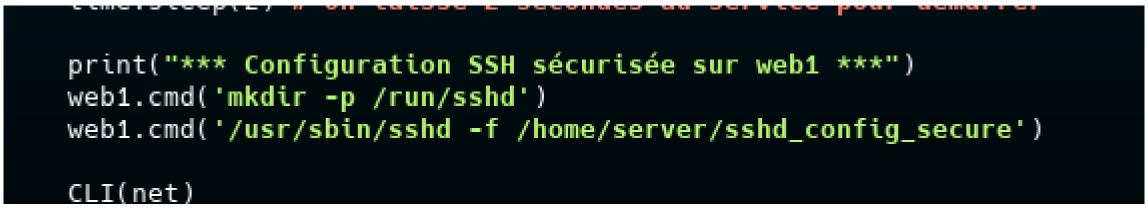
```
mininet> admin cat /root/.ssh/id_rsa.pub > authorized_keys_temp
mininet> web1 mkdir -p /root/.ssh
mininet> web1 cp authorized_keys_temp /root/.ssh/authorized_keys
```

3. Configurer le serveur SSH sur web1 pour refuser les mots de passe



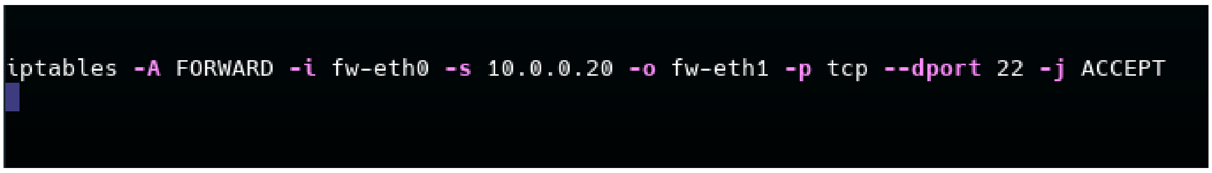
```
server@srv ~  (0.019s)
cat sshd_config_secure

PermitRootLogin yes
PubkeyAuthentication yes
PasswordAuthentication no
ChallengeResponseAuthentication no
UsePAM yes
PrintMotd no
AcceptEnv LANG LC_*
Subsystem sftp /usr/lib/openssh/sftp-server
```

4. Appliquer la config dans projet_topo.py



```
print("*** Configuration SSH sécurisée sur web1 ***")
web1.cmd('mkdir -p /run/sshd')
web1.cmd('/usr/sbin/sshd -f /home/server/sshd_config_secure')

CLI(net)
```

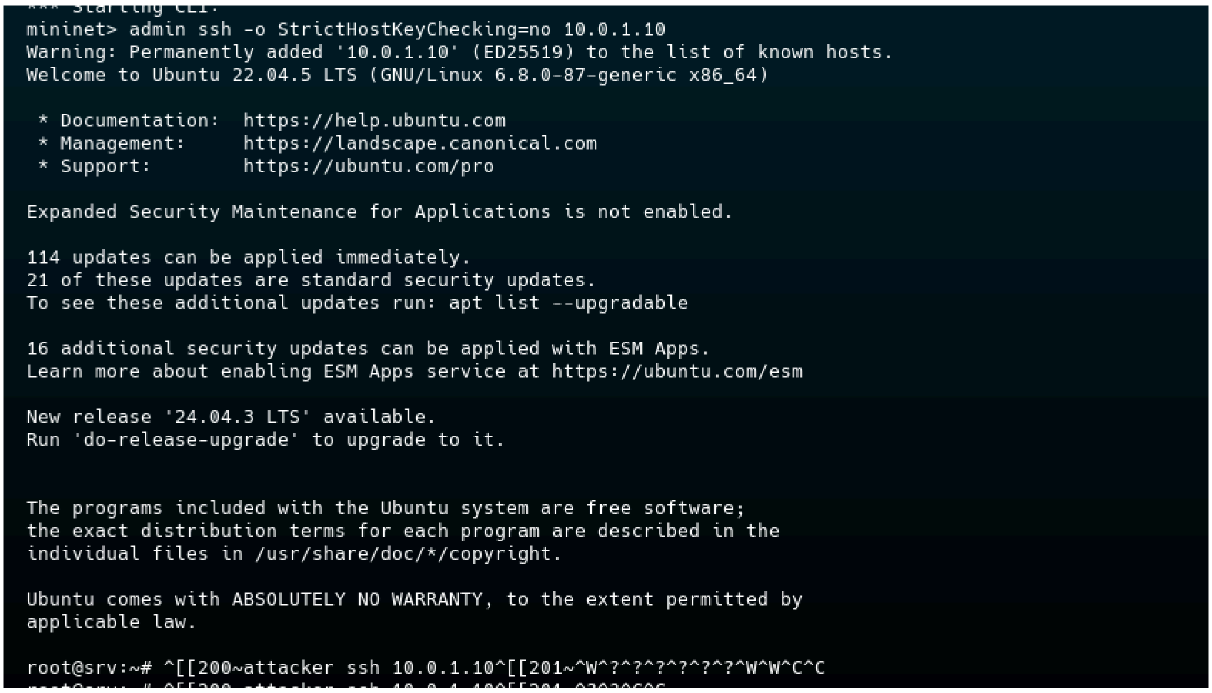
5. Mettre à jour le Firewall



```
iptables -A FORWARD -i fw-eth0 -s 10.0.0.20 -o fw-eth1 -p tcp --dport 22 -j ACCEPT
```

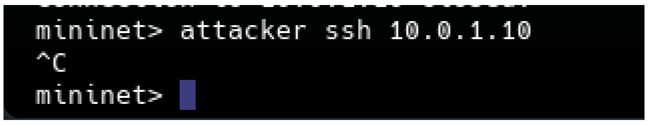
Étape 9 : Test de l'administration

Test Admin (autorisé) : `admin ssh -o StrictHostKeyChecking=no 10.0.1.10` ***Tu devrais entrer sans mot de passe.***



```
mininet> admin ssh -o StrictHostKeyChecking=no 10.0.1.10
Warning: Permanently added '10.0.1.10' (ED25519) to the list of known hosts.
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 6.8.0-87-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

Expanded Security Maintenance for Applications is not enabled.

114 updates can be applied immediately.
21 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

16 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

New release '24.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

root@srv:~# ^[[200~attacker ssh 10.0.1.10^[[201~^W^?^?^?^?^?^?^W^W^C^C
```

Test Attacker (interdit)

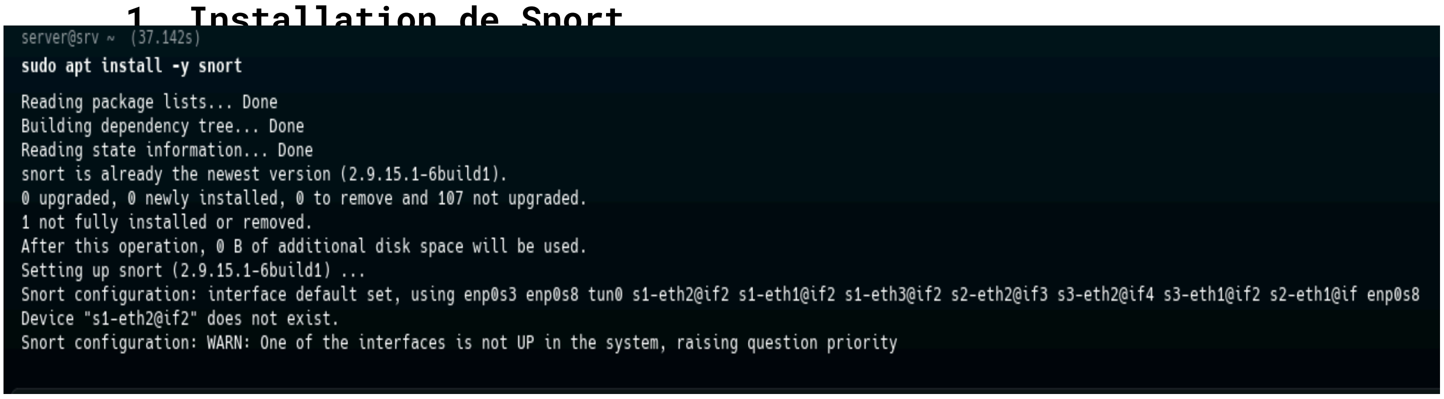


```
mininet> attacker ssh 10.0.1.10
^C
mininet>
```

La commande doit rester "figée" (bloquée par le pare-feu) ou afficher `Connection timed out`. C'est la preuve que ton Zero Trust fonctionne : l'admin passe, l'attaquant est bloqué.

Étape 10 : Détection d'Intrusion avec Snort (Point 5.5)

1. Installation de Snort

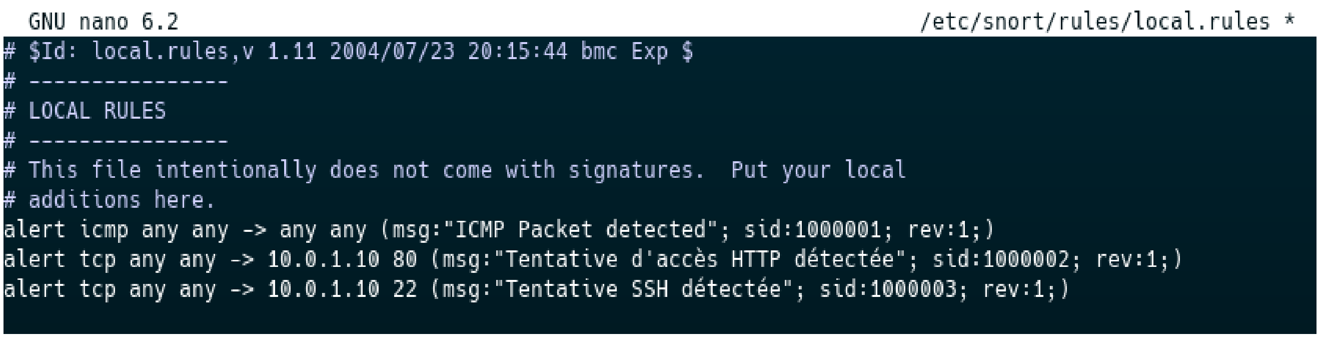


```
server@srv ~ (37.142s)
sudo apt install -y snort

Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
snort is already the newest version (2.9.15.1-6build1).
0 upgraded, 0 newly installed, 0 to remove and 107 not upgraded.
1 not fully installed or removed.
After this operation, 0 B of additional disk space will be used.
Setting up snort (2.9.15.1-6build1) ...
Snort configuration: interface default set, using enp0s3 enp0s8 tun0 s1-eth2@if2 s1-eth1@if2 s1-eth3@if2 s2-eth2@if3 s3-eth2@if4 s3-eth1@if2 s2-eth1@if enp0s8
Device "s1-eth2@if2" does not exist.
Snort configuration: WARN: One of the interfaces is not UP in the system, raising question priority
```

2. Création d'une règle personnalisée:

Nous voulons que Snort nous alerte si quelqu'un essaie de scanner le réseau (Nmap) ou d'attaquer le serveur web.

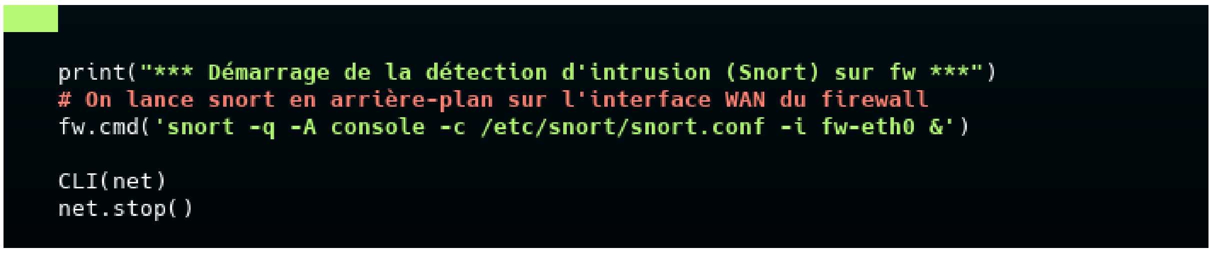
```
sudo nano /etc/snort/rules/local.rules
```



```
  GNU nano 6.2                                         /etc/snort/rules/local.rules *
# $Id: local.rules,v 1.11 2004/07/23 20:15:44 bmc Exp $
# ----------------
# LOCAL RULES
# ----------------
# This file intentionally does not come with signatures.  Put your local
# additions here.
alert icmp any any -> any any (msg:"ICMP Packet detected"; sid:1000001; rev:1;)
alert tcp any any -> 10.0.1.10 80 (msg:"Tentative d'accès HTTP détectée"; sid:1000002; rev:1;)
alert tcp any any -> 10.0.1.10 22 (msg:"Tentative SSH détectée"; sid:1000003; rev:1;)
```

3. Lancer Snort sur le Firewall (`fw`)

projet_topo.py

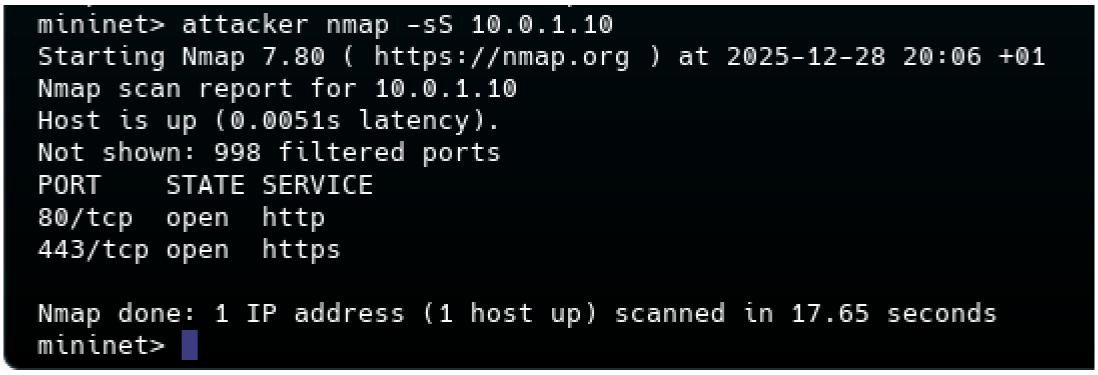


```
print("*** Démarrage de la détection d'intrusion (Snort) sur fw ***")
# On lance snort en arrière-plan sur l'interface WAN du firewall
fw.cmd('snort -q -A console -c /etc/snort/snort.conf -i fw-eth0 &')

CLI(net)
net.stop()
```

Étape 11 : Le Scénario d'Attaque (Preuve pour ton rapport)

Pendant que l'attaquant lance son scan, tu dois observer deux choses :

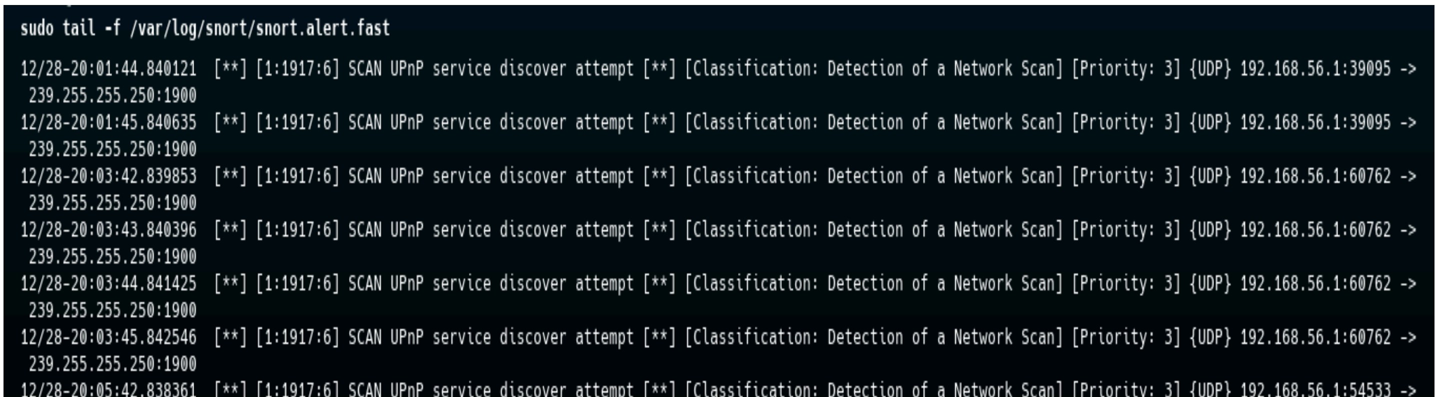
1. **Le Pare-feu bloque :** Nmap devrait t'afficher que les ports sont `filtered`. C'est la preuve que ton script `firewall.sh` fait son travail.



```
mininet> attacker nmap -sS 10.0.1.10
Starting Nmap 7.80 ( https://nmap.org ) at 2025-12-28 20:06 +01
Nmap scan report for 10.0.1.10
Host is up (0.0051s latency).
Not shown: 998 filtered ports
PORT    STATE SERVICE
80/tcp  open  http
443/tcp open  https

Nmap done: 1 IP address (1 host up) scanned in 17.65 seconds
mininet>
```

"998 filtered ports" : Cela signifie que sur 1000 ports testés, 998 ont été totalement ignorés par ton Pare-feu. L'attaquant ne sait même pas si ces ports existent. C'est le comportement attendu d'un **Stateful Firewall**.
"80/tcp open" et "443/tcp open" : Ce sont les seuls ports que tu as autorisés dans ton script `firewall.sh`. Nmap les voit "ouverts" car ton serveur Nginx répond derrière.
Conclusion : Ta segmentation réseau est parfaite. L'attaquant est limité à ce que tu as explicitement exposé (la DMZ).

2. **Snort alerte :** Si Snort est bien lancé sur `fw`, il doit "voir" passer les paquets de Nmap avant qu'ils ne soient jetés par le pare-feu.



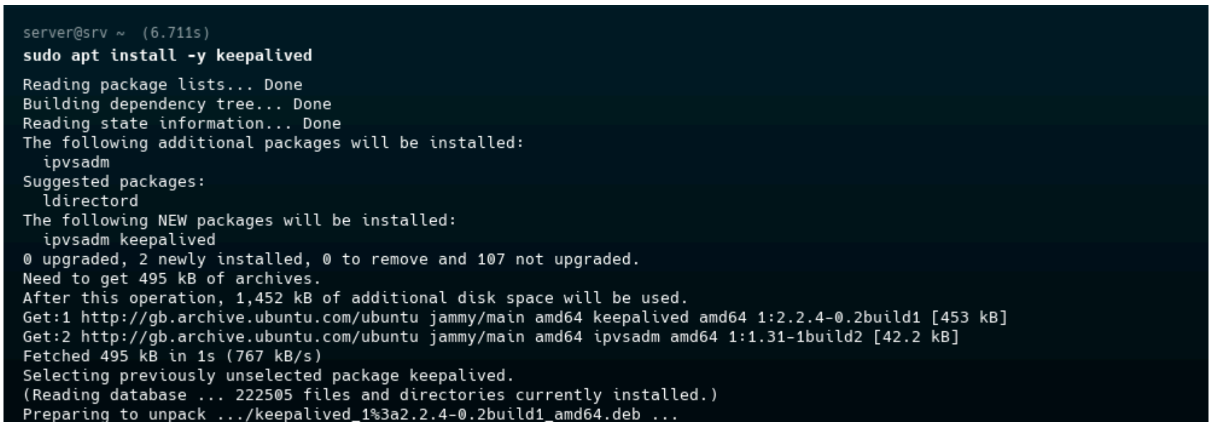
```
sudo tail -f /var/log/snort/snort.alert.fast

12/28-20:01:44.840121  [**] [1:1917:6] SCAN UPnP service discover attempt [**] [Classification: Detection of a Network Scan] [Priority: 3] {UDP} 192.168.56.1:39095 ->
 239.255.255.250:1900
12/28-20:01:45.840635  [**] [1:1917:6] SCAN UPnP service discover attempt [**] [Classification: Detection of a Network Scan] [Priority: 3] {UDP} 192.168.56.1:39095 ->
 239.255.255.250:1900
12/28-20:03:42.839853  [**] [1:1917:6] SCAN UPnP service discover attempt [**] [Classification: Detection of a Network Scan] [Priority: 3] {UDP} 192.168.56.1:60762 ->
 239.255.255.250:1900
12/28-20:03:43.840396  [**] [1:1917:6] SCAN UPnP service discover attempt [**] [Classification: Detection of a Network Scan] [Priority: 3] {UDP} 192.168.56.1:60762 ->
 239.255.255.250:1900
12/28-20:03:44.841425  [**] [1:1917:6] SCAN UPnP service discover attempt [**] [Classification: Detection of a Network Scan] [Priority: 3] {UDP} 192.168.56.1:60762 ->
 239.255.255.250:1900
12/28-20:03:45.842546  [**] [1:1917:6] SCAN UPnP service discover attempt [**] [Classification: Detection of a Network Scan] [Priority: 3] {UDP} 192.168.56.1:60762 ->
 239.255.255.250:1900
12/28-20:05:42.838361  [**] [1:1917:6] SCAN UPnP service discover attempt [**] [Classification: Detection of a Network Scan] [Priority: 3] {UDP} 192.168.56.1:54533 ->
```

Étape 12 : La Haute Disponibilité (Le Cluster Actif/Passif)

Le Concept

1. On remplace le nœud `fw` par deux nœuds : `fw1` (Maître) et `fw2` (Esclave).
2. Ils partagent une **IP Virtuelle (VIP)** : `10.0.0.1`.
3. Si `fw1` tombe, `fw2` détecte l'absence de "battement de cœur" (Heartbeat) et s'attribue l'IP `10.0.0.1`.
4. Pour l'utilisateur (l'attaquant ou l'admin), le service continue sans interruption.

1. Préparation des outils

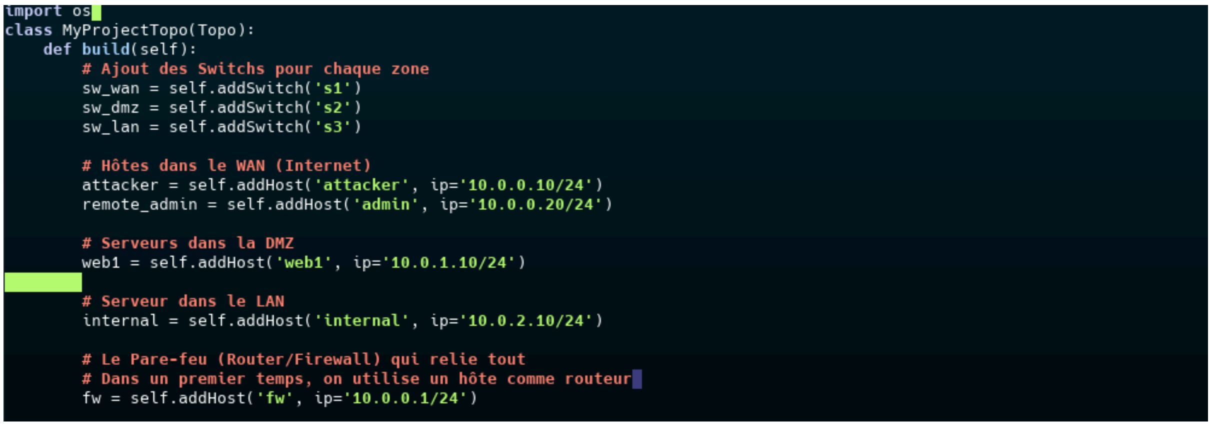


```
server@srv ~  (6.711s)
sudo apt install -y keepalived
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  ipvsadm
Suggested packages:
  ldirectord
The following NEW packages will be installed:
  ipvsadm keepalived
0 upgraded, 2 newly installed, 0 to remove and 107 not upgraded.
Need to get 495 kB of archives.
After this operation, 1,452 kB of additional disk space will be used.
Get:1 http://gb.archive.ubuntu.com/ubuntu jammy/main amd64 keepalived amd64 1:2.2.4-0.2build1 [453 kB]
Get:2 http://gb.archive.ubuntu.com/ubuntu jammy/main amd64 ipvsadm amd64 1:1.31-1build2 [42.2 kB]
Fetched 495 kB in 1s (767 kB/s)
Selecting previously unselected package keepalived.
(Reading database ... 222505 files and directories currently installed.)
Preparing to unpack .../keepalived_1%3a2.2.4-0.2build1_amd64.deb ...
```

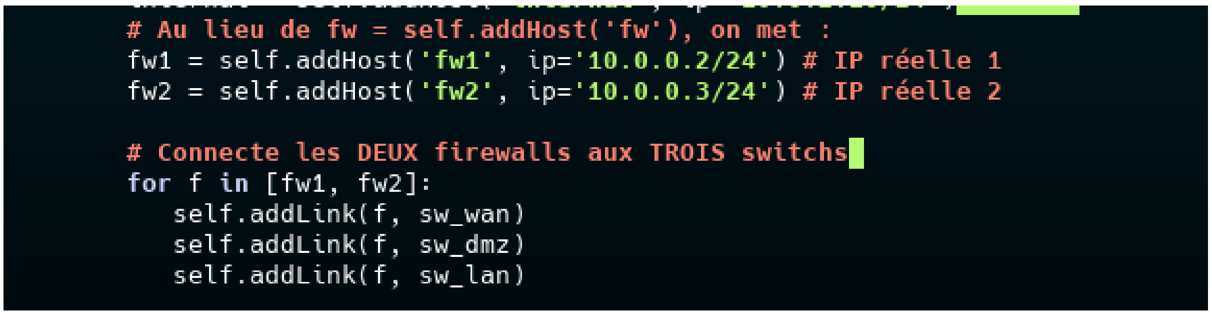
2. Modification de la Topologie (`projet_topo.py`)

Tu dois modifier ton script pour ajouter le deuxième Firewall. Voici les changements clés à faire dans la classe `MyProjectTopo` :

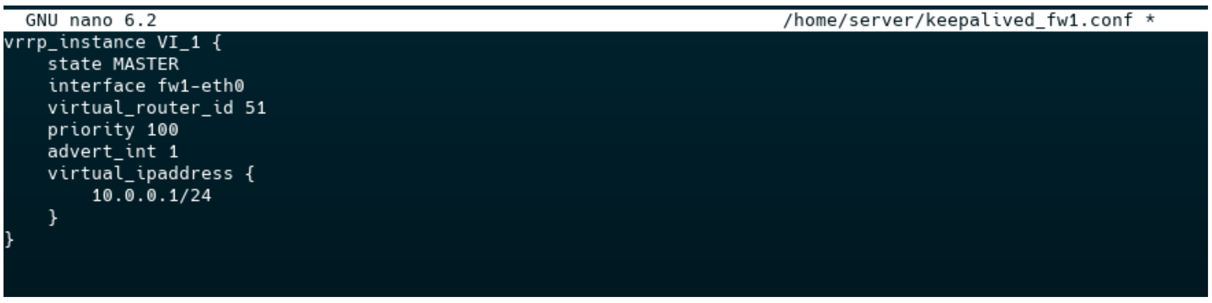
Avant on avait un seulle fw:

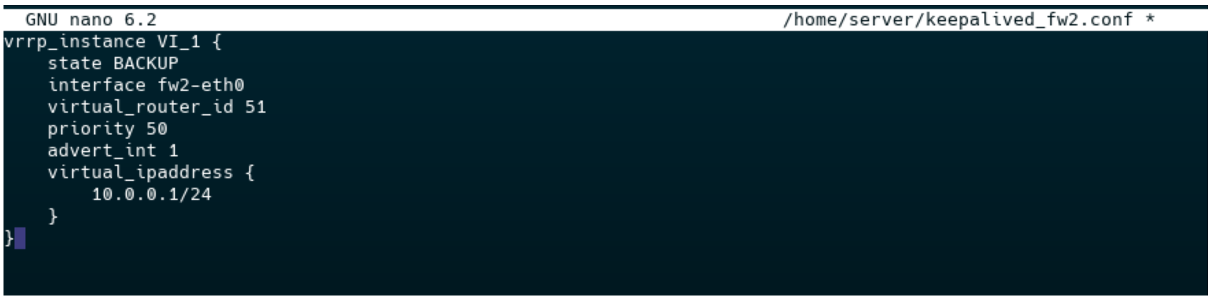

```
import os
class MyProjectTopo(Topo):
    def build(self):
        # Ajout des Switchs pour chaque zone
        sw_wan = self.addSwitch('s1')
        sw_dmz = self.addSwitch('s2')
        sw_lan = self.addSwitch('s3')

        # Hôtes dans le WAN (Internet)
        attacker = self.addHost('attacker', ip='10.0.0.10/24')
        remote_admin = self.addHost('admin', ip='10.0.0.20/24')

        # Serveurs dans la DMZ
        web1 = self.addHost('web1', ip='10.0.1.10/24')

        # Serveur dans le LAN
        internal = self.addHost('internal', ip='10.0.2.10/24')

        # Le Pare-feu (Router/Firewall) qui relie tout
        # Dans un premier temps, on utilise un hôte comme routeur
        fw = self.addHost('fw', ip='10.0.0.1/24')
```

maintenant on ajoute le deuxieme :

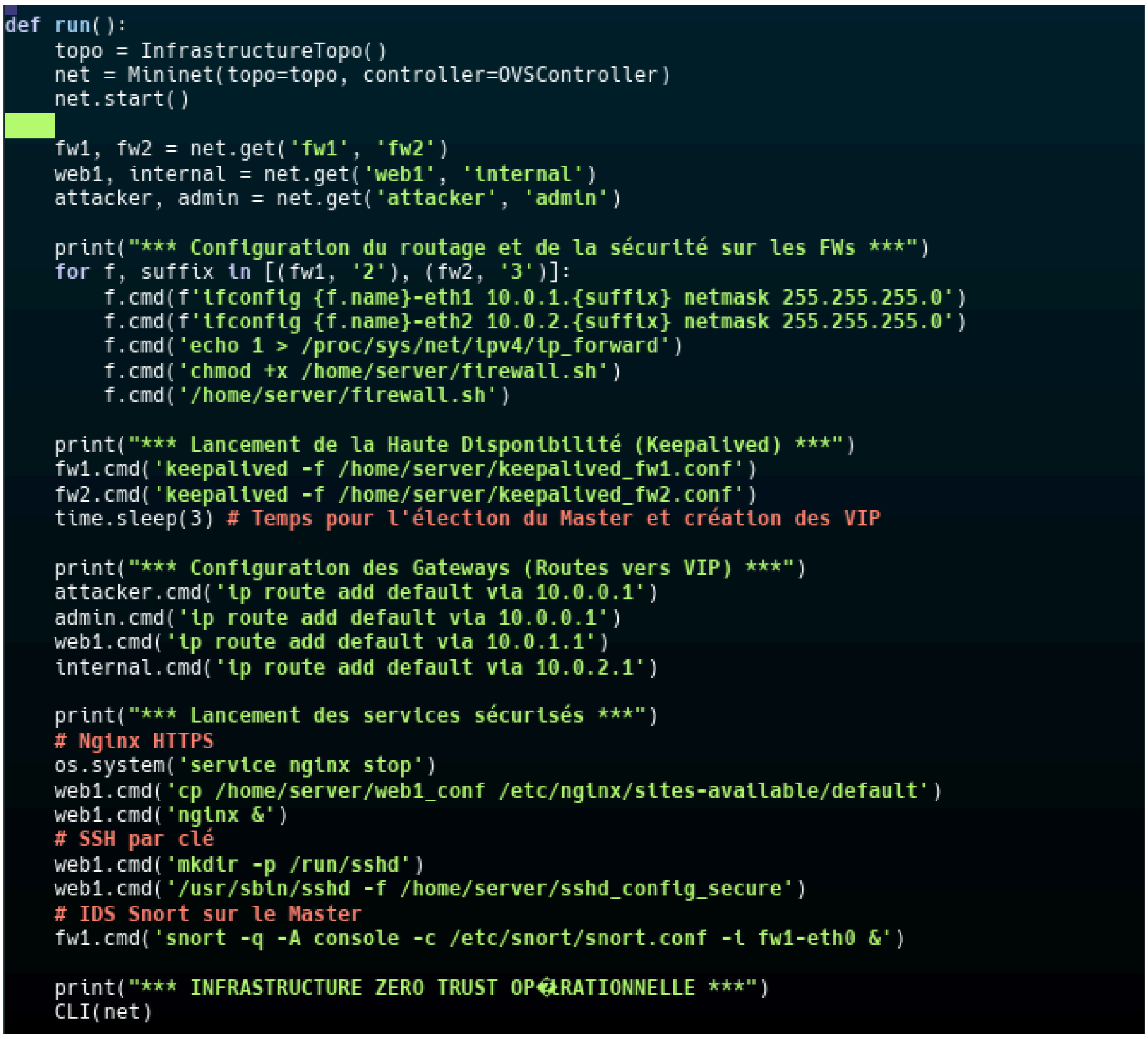

```
        # Au lieu de fw = self.addHost('fw'), on met :
        fw1 = self.addHost('fw1', ip='10.0.0.2/24') # IP réelle 1
        fw2 = self.addHost('fw2', ip='10.0.0.3/24') # IP réelle 2

        # Connecte les DEUX firewalls aux TROIS switchs
        for f in [fw1, fw2]:
            self.addLink(f, sw_wan)
            self.addLink(f, sw_dmz)
            self.addLink(f, sw_lan)
```

3. Configuration de Keepalived


```
  GNU nano 6.2                                         /home/server/keepalived_fw1.conf *
vrrp_instance VI_1 {
    state MASTER
    interface fw1-eth0
    virtual_router_id 51
    priority 100
    advert_int 1
    virtual_ipaddress {
        10.0.0.1/24
    }
}
```


```
  GNU nano 6.2                                         /home/server/keepalived_fw2.conf *
vrrp_instance VI_1 {
    state BACKUP
    interface fw2-eth0
    virtual_router_id 51
    priority 50
    advert_int 1
    virtual_ipaddress {
        10.0.0.1/24
    }
}
```

4. Automatisation dans le script Python



```python
def run():
    topo = InfrastructureTopo()
    net = Mininet(topo=topo, controller=OVSController)
    net.start()

    fw1, fw2 = net.get('fw1', 'fw2')
    web1, internal = net.get('web1', 'internal')
    attacker, admin = net.get('attacker', 'admin')

    print("*** Configuration du routage et de la sécurité sur les FWs ***")
    for f, suffix in [(fw1, '2'), (fw2, '3')]:
        f.cmd(f'ifconfig {f.name}-eth1 10.0.1.{suffix} netmask 255.255.255.0')
        f.cmd(f'ifconfig {f.name}-eth2 10.0.2.{suffix} netmask 255.255.255.0')
        f.cmd('echo 1 > /proc/sys/net/ipv4/ip_forward')
        f.cmd('chmod +x /home/server/firewall.sh')
        f.cmd('/home/server/firewall.sh')

    print("*** Lancement de la Haute Disponibilité (Keepalived) ***")
    fw1.cmd('keepalived -f /home/server/keepalived_fw1.conf')
    fw2.cmd('keepalived -f /home/server/keepalived_fw2.conf')
    time.sleep(3) # Temps pour l'élection du Master et création des VIP

    print("*** Configuration des Gateways (Routes vers VIP) ***")
    attacker.cmd('ip route add default via 10.0.0.1')
    admin.cmd('ip route add default via 10.0.0.1')
    web1.cmd('ip route add default via 10.0.1.1')
    internal.cmd('ip route add default via 10.0.2.1')

    print("*** Lancement des services sécurisés ***")
    # Nginx HTTPS
    os.system('service nginx stop')
    web1.cmd('cp /home/server/web1_conf /etc/nginx/sites-available/default')
    web1.cmd('nginx &')
    # SSH par clé
    web1.cmd('mkdir -p /run/sshd')
    web1.cmd('/usr/sbin/sshd -f /home/server/sshd_config_secure')
    # IDS Snort sur le Master
    fw1.cmd('snort -q -A console -c /etc/snort/snort.conf -i fw1-eth0 &')

    print("*** INFRASTRUCTURE ZERO TRUST OP�RATIONNELLE ***")
    CLI(net)
```

Dans la fonction `run()`, configure le routage et lance Keepalived sur les deux :

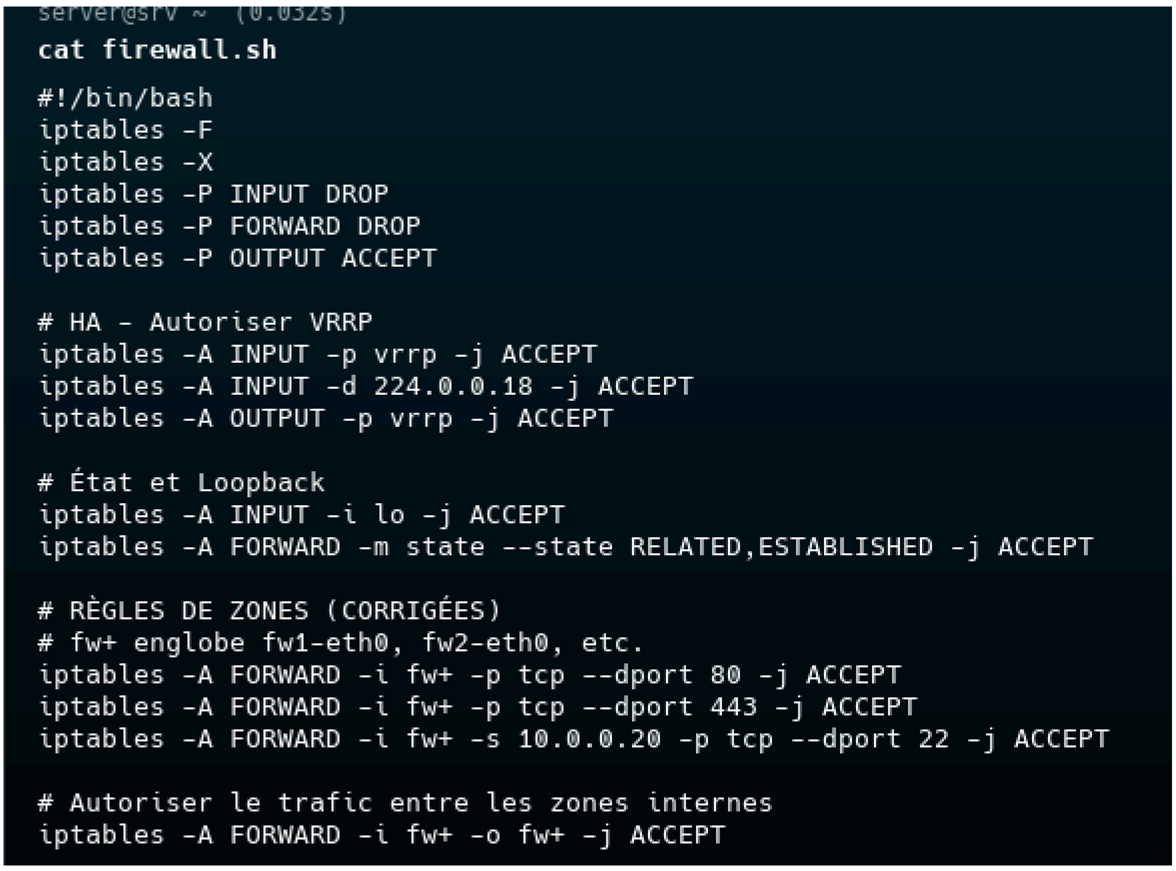
2. Comparaison : Ce qui change dans `run()`

Élément	Ancienne version (1 FW)	Nouvelle version (HA - 2 FWs)
Objets	`fw = net.get('fw')`	`fw1, fw2 = net.get('fw1', 'fw2')`
IP Interfaces	`fw` a `10.0.1.1`	`fw1` a `10.0.1.2` / `fw2` a `10.0.1.3`

Gestion VIP	Manuelle (ifconfig)	Automatique via **Keepalived**
Passerelles Hôtes	Pointent vers fw	Pointent vers la **VIP** (gérée par Keepalived)

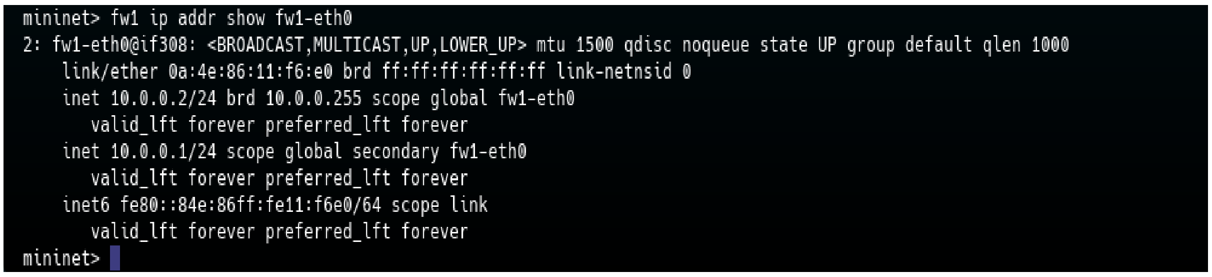
5. Le Test de Validation (Scénario de panne)

Étape 1 : Autoriser Keepalived dans `firewall.sh`

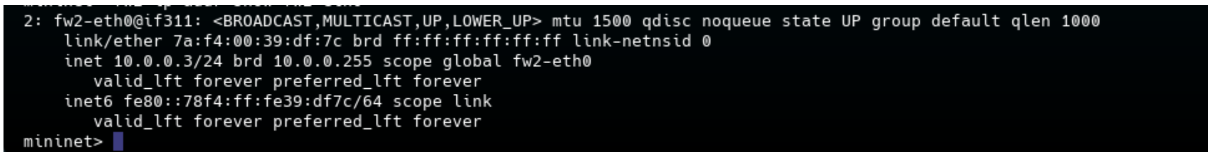


```
server@srv ~  (0.032s)
cat firewall.sh
#!/bin/bash
iptables -F
iptables -X
iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -P OUTPUT ACCEPT

# HA - Autoriser VRRP
iptables -A INPUT -p vrrp -j ACCEPT
iptables -A INPUT -d 224.0.0.18 -j ACCEPT
iptables -A OUTPUT -p vrrp -j ACCEPT

# État et Loopback
iptables -A INPUT -i lo -j ACCEPT
iptables -A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT

# RÈGLES DE ZONES (CORRIGÉES)
# fw+ englobe fw1-eth0, fw2-eth0, etc.
iptables -A FORWARD -i fw+ -p tcp --dport 80 -j ACCEPT
iptables -A FORWARD -i fw+ -p tcp --dport 443 -j ACCEPT
iptables -A FORWARD -i fw+ -s 10.0.0.20 -p tcp --dport 22 -j ACCEPT

# Autoriser le trafic entre les zones internes
iptables -A FORWARD -i fw+ -o fw+ -j ACCEPT
```

Verification des addresses ip avant la panne:

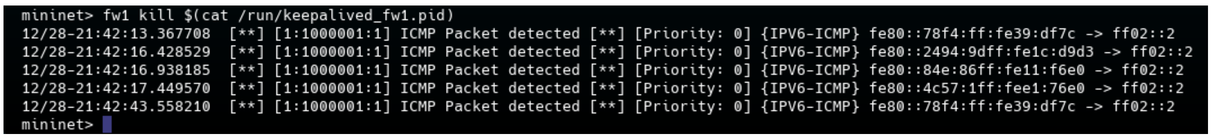


```
mininet> fw1 ip addr show fw1-eth0
2: fw1-eth0@if308: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 0a:4e:86:11:f6:e0 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.0.0.2/24 brd 10.0.0.255 scope global fw1-eth0
       valid_lft forever preferred_lft forever
    inet 10.0.0.1/24 scope global secondary fw1-eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::84e:86ff:fe11:f6e0/64 scope link
       valid_lft forever preferred_lft forever
mininet>
```



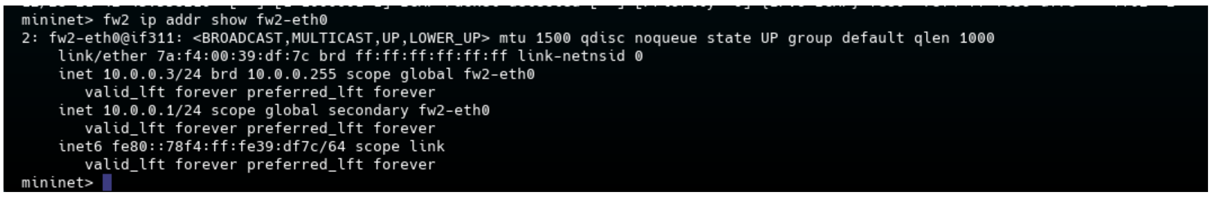
```
2: fw2-eth0@if311: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 7a:f4:00:39:df:7c brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.0.0.3/24 brd 10.0.0.255 scope global fw2-eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::78f4:ff:fe39:df7c/64 scope link
       valid_lft forever preferred_lft forever
mininet>
```

En mettant fw1 en panne :

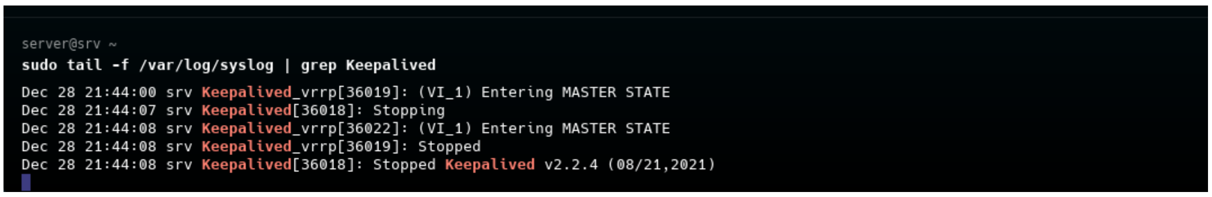


```
mininet> fw1 kill $(cat /run/keepalived_fw1.pid)
12/28-21:42:13.367708  [**] [1:1000001:1] ICMP Packet detected [**] [Priority: 0] {IPV6-ICMP} fe80::78f4:ff:fe39:df7c -> ff02::2
12/28-21:42:16.428529  [**] [1:1000001:1] ICMP Packet detected [**] [Priority: 0] {IPV6-ICMP} fe80::2494:9dff:fe1c:d9d3 -> ff02::2
12/28-21:42:16.938185  [**] [1:1000001:1] ICMP Packet detected [**] [Priority: 0] {IPV6-ICMP} fe80::84e:86ff:fe11:f6e0 -> ff02::2
12/28-21:42:17.449570  [**] [1:1000001:1] ICMP Packet detected [**] [Priority: 0] {IPV6-ICMP} fe80::4c57:1ff:fee1:76e0 -> ff02::2
12/28-21:42:43.558210  [**] [1:1000001:1] ICMP Packet detected [**] [Priority: 0] {IPV6-ICMP} fe80::78f4:ff:fe39:df7c -> ff02::2
mininet>
```

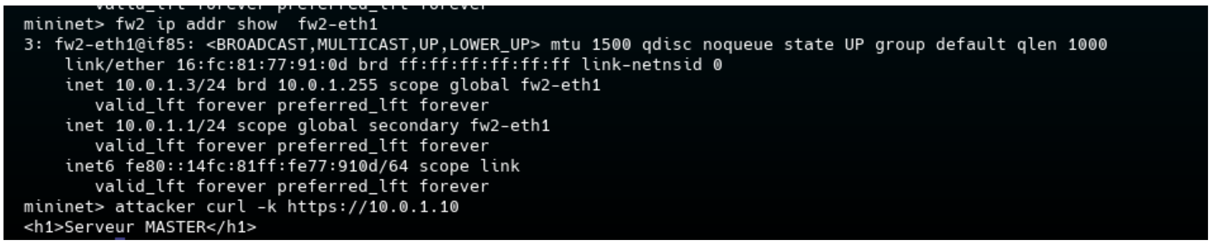
En verifiant le basculement:



```
mininet> fw2 ip addr show fw2-eth0
2: fw2-eth0@if311: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 7a:f4:00:39:df:7c brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.0.0.3/24 brd 10.0.0.255 scope global fw2-eth0
       valid_lft forever preferred_lft forever
    inet 10.0.0.1/24 scope global secondary fw2-eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::78f4:ff:fe39:df7c/64 scope link
       valid_lft forever preferred_lft forever
mininet>
```



```
server@srv ~
sudo tail -f /var/log/syslog | grep Keepalived
Dec 28 21:44:00 srv Keepalived_vrrp[36019]: (VI_1) Entering MASTER STATE
Dec 28 21:44:07 srv Keepalived[36018]: Stopping
Dec 28 21:44:08 srv Keepalived_vrrp[36022]: (VI_1) Entering MASTER STATE
Dec 28 21:44:08 srv Keepalived_vrrp[36019]: Stopped
Dec 28 21:44:08 srv Keepalived[36018]: Stopped Keepalived v2.2.4 (08/21,2021)
```

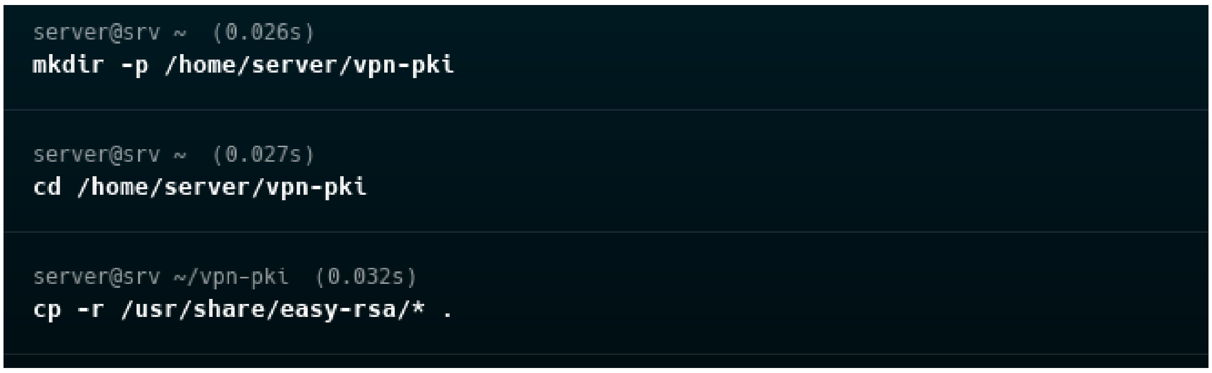


```
mininet> fw2 ip addr show  fw2-eth1
3: fw2-eth1@if85: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 16:fc:81:77:91:0d brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.0.1.3/24 brd 10.0.1.255 scope global fw2-eth1
       valid_lft forever preferred_lft forever
    inet 10.0.1.1/24 scope global secondary fw2-eth1
       valid_lft forever preferred_lft forever
    inet6 fe80::14fc:81ff:fe77:910d/64 scope link
       valid_lft forever preferred_lft forever
mininet> attacker curl -k https://10.0.1.10
<h1>Serveur MASTER</h1>
```

OPENVPN:

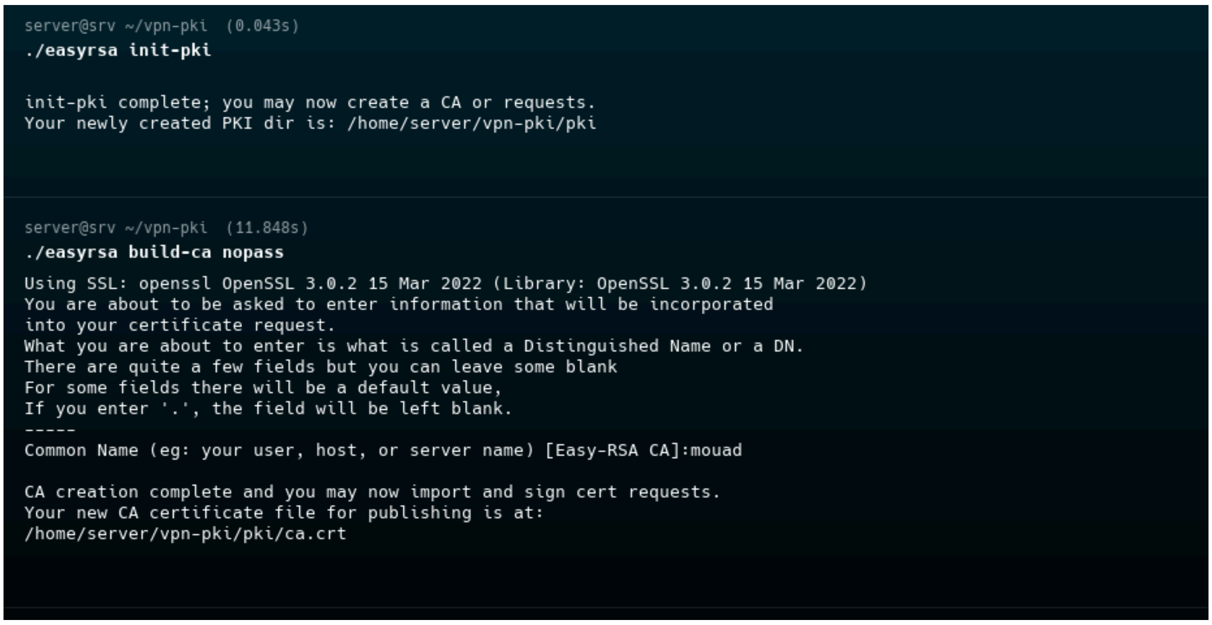
Étape 1 : Installation et Génération des Clés (Sur ta VM)

Crée un dossier pour tes clés :



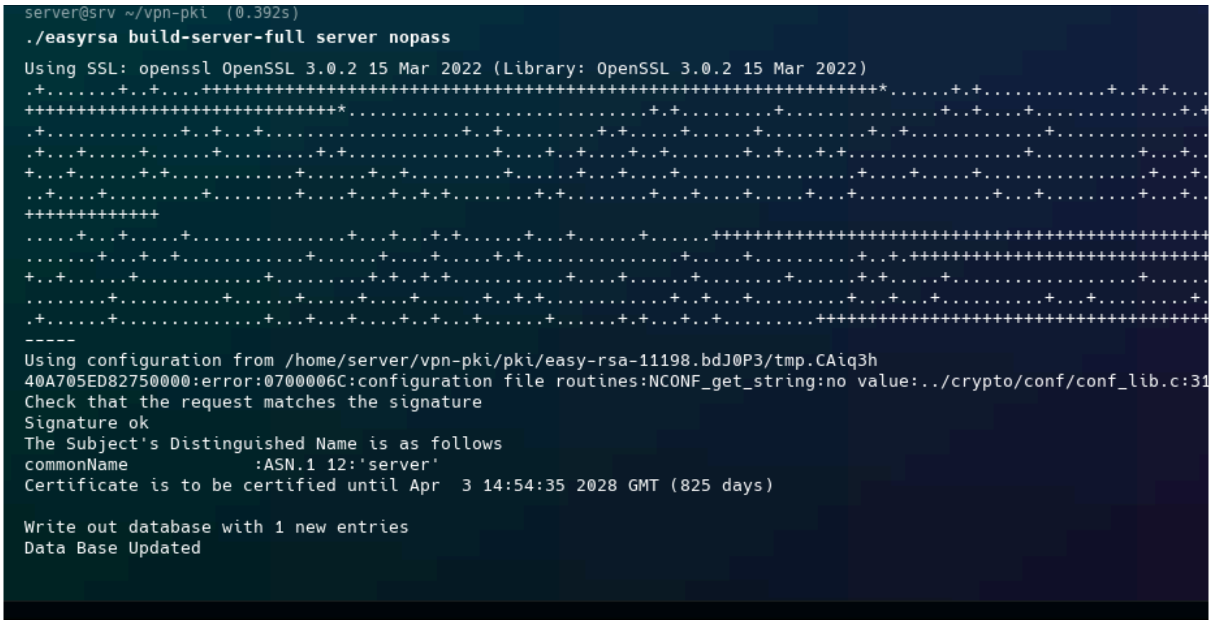
```
server@srv ~  (0.026s)
mkdir -p /home/server/vpn-pki

server@srv ~  (0.027s)
cd /home/server/vpn-pki

server@srv ~/vpn-pki  (0.032s)
cp -r /usr/share/easy-rsa/* .
```

Génère les clés (Copie-colle ces commandes une par une) : ***Cela crée une Autorité de Certification (CA), un certificat pour le Serveur et un pour le Client.***

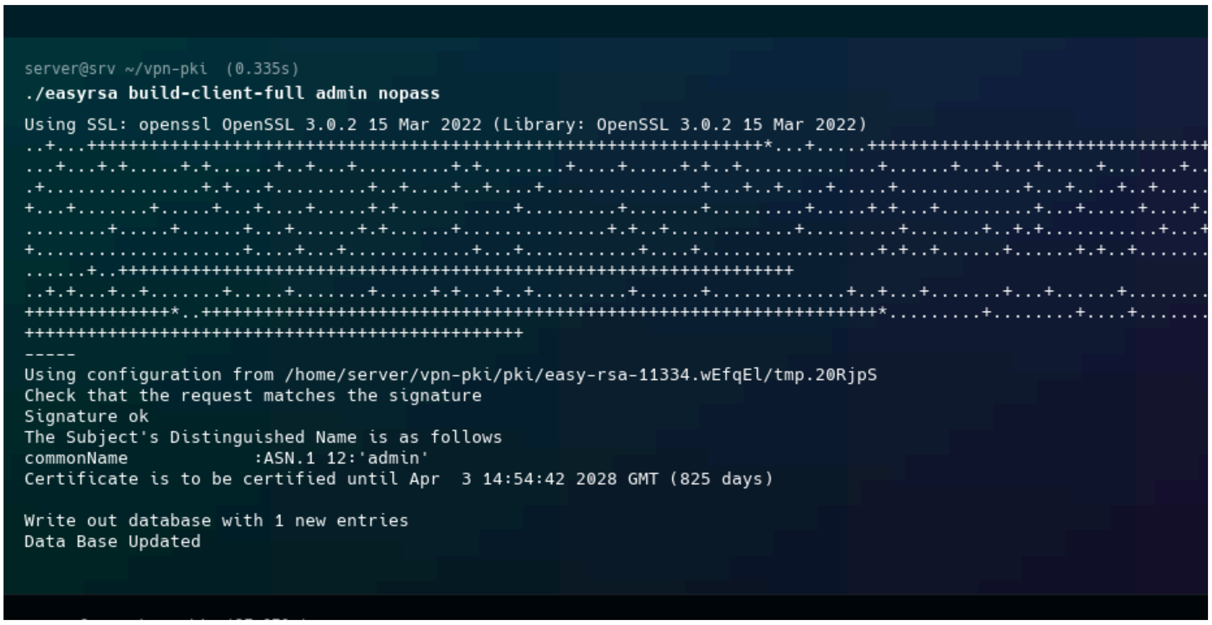


```
server@srv ~/vpn-pki  (0.043s)
./easyrsa init-pki

init-pki complete; you may now create a CA or requests.
Your newly created PKI dir is: /home/server/vpn-pki/pki
```

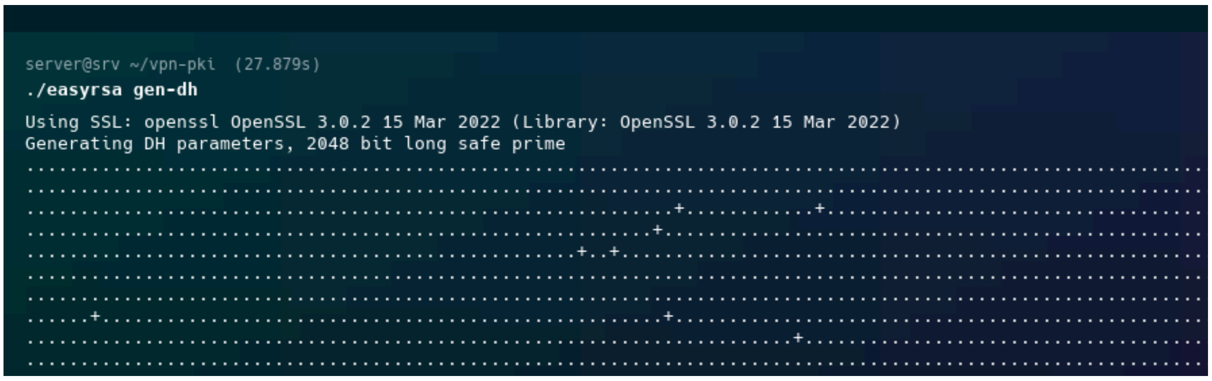
```
server@srv ~/vpn-pki  (11.848s)
./easyrsa build-ca nopass
Using SSL: openssl OpenSSL 3.0.2 15 Mar 2022 (Library: OpenSSL 3.0.2 15 Mar 2022)
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Common Name (eg: your user, host, or server name) [Easy-RSA CA]:mouad

CA creation complete and you may now import and sign cert requests.
Your new CA certificate file for publishing is at:
/home/server/vpn-pki/pki/ca.crt
```



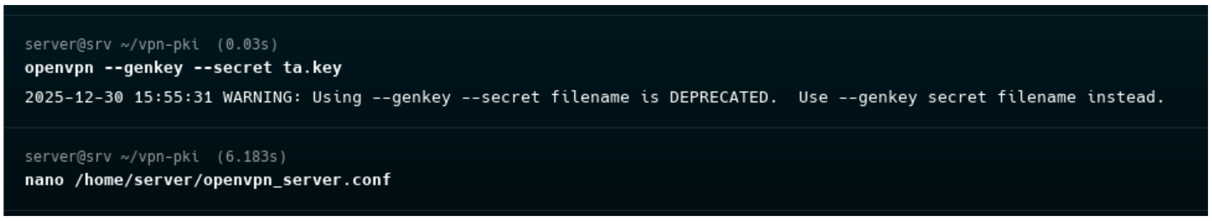
```
server@srv ~/vpn-pki  (0.392s)
./easyrsa build-server-full server nopass
Using SSL: openssl OpenSSL 3.0.2 15 Mar 2022 (Library: OpenSSL 3.0.2 15 Mar 2022)
.+.......+..+....+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*......+.+............+..+.+....
+++++++++++++++++++++++++++++++*.............................+.+.........+...............+..+....+..............+.+
.+............+..+...+...................+..+.........+.+.....+......+..........+..+.............+...............
.+...+.....+......+.........+.+.............+....+..+....+..+.......+..+...+.+.................+..........+...+..
+...+......+.+............+......+..+.........+......+...+....+.................+....+.....+................+...+.
..+....+.........+........+....+...+..+.+........+.+........+...+....+.....+...+............+...+.........+...+..
+++++++++++++
.....+...+.....+..............+...+...+.+......+...+......+......++++++++++++++++++++++++++++++++++++++++++++++++
.......+...+..+............+......+....+.....+.+..............+.....+..........+..+.++++++++++++++++++++++++++++
+..+......+............+.........+.+..+.+...........+....+......+.........+......+.+.....+.................+.....
........+..........+......+.....+....+......+..+.+............+..+...+.........+...+...+..........+...+.........+.
.+......+.............+...+...+....+..+...+......+......+.+...+..+.........++++++++++++++++++++++++++++++++++++++
-----
Using configuration from /home/server/vpn-pki/pki/easy-rsa-11198.bdJ0P3/tmp.CAiq3h
40A705ED82750000:error:0700006C:configuration file routines:NCONF_get_string:no value:../crypto/conf/conf_lib.c:31
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName            :ASN.1 12:'server'
Certificate is to be certified until Apr  3 14:54:35 2028 GMT (825 days)

Write out database with 1 new entries
Data Base Updated
```



```
server@srv ~/vpn-pki  (0.335s)
./easyrsa build-client-full admin nopass
Using SSL: openssl OpenSSL 3.0.2 15 Mar 2022 (Library: OpenSSL 3.0.2 15 Mar 2022)
..+...++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*...+.....++++++++++++++++++++++++++++++++
...+...+.+.....+.+......+..+...+.........+.+........+....+.....+.+..+.............+......+...+...+.....+.......+..
.+...............+.+...+.........+..+....+..+....+..............+...+..+....+.....+............+...+....+..+.....
+...+.......+.....+...+....+.....+.+...........+.........+.......+.........+.....+.+...+.........+...+.....+....+.
........+.....+......+...+......+.+......+..............+.+..+............+.........+.......+..+.+..........+...+
+....................+....+...+...........+..+..........+....+..................+.+..+......+......+.+..+......
......+..++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
..+.+...+..+.......+.....+.......+.....+.+...+..+.........+......+............+..+...+.......+...+......+.......
++++++++++++++*..++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*.........+........+....+......
++++++++++++++++++++++++++++++++++++++++++++++++
-----
Using configuration from /home/server/vpn-pki/pki/easy-rsa-11334.wEfqEl/tmp.20RjpS
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName            :ASN.1 12:'admin'
Certificate is to be certified until Apr  3 14:54:42 2028 GMT (825 days)

Write out database with 1 new entries
Data Base Updated
```

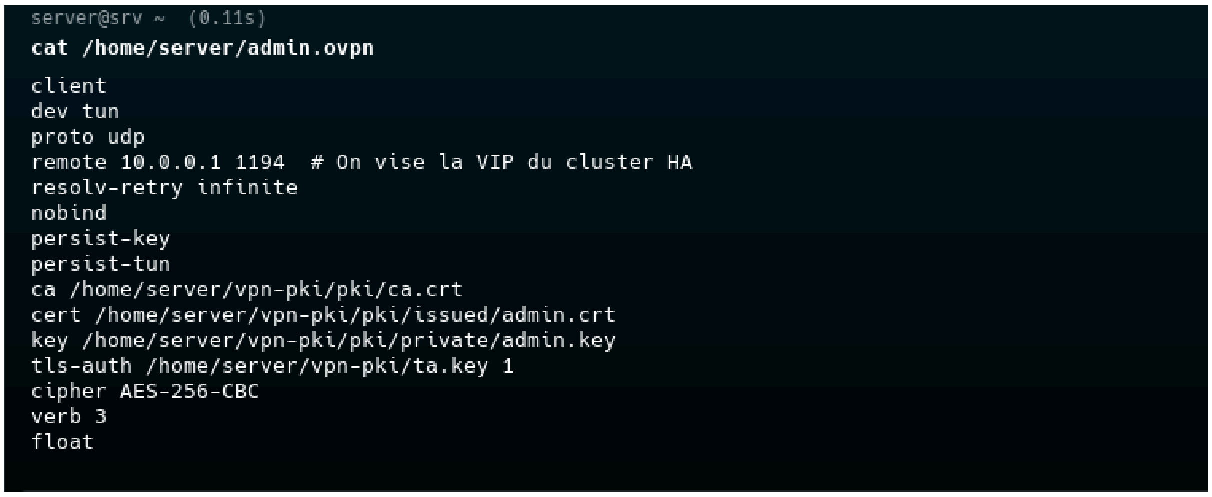


```
server@srv ~/vpn-pki  (27.879s)
./easyrsa gen-dh
Using SSL: openssl OpenSSL 3.0.2 15 Mar 2022 (Library: OpenSSL 3.0.2 15 Mar 2022)
Generating DH parameters, 2048 bit long safe prime
..................................................................................................................
..................................................................................................................
..............................................................+.............+...................................
............................................................+.....................................................
.....................................................+..+.........................................................
..................................................................................................................
..................................................................................................................
......+.......................................................+...................................................
..........................................................................+.......................................
..................................................................................................................
```



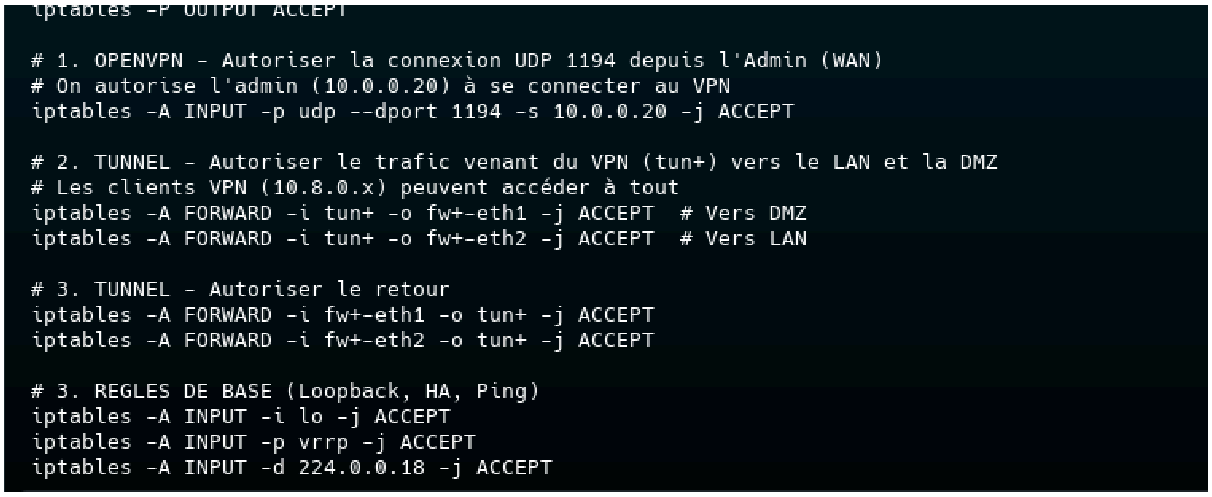
```
server@srv ~/vpn-pki  (0.03s)
openvpn --genkey --secret ta.key
2025-12-30 15:55:31 WARNING: Using --genkey --secret filename is DEPRECATED.  Use --genkey secret filename instead.

server@srv ~/vpn-pki  (6.183s)
nano /home/server/openvpn_server.conf
```

Étape 2 : Création du fichier de configuration Serveur



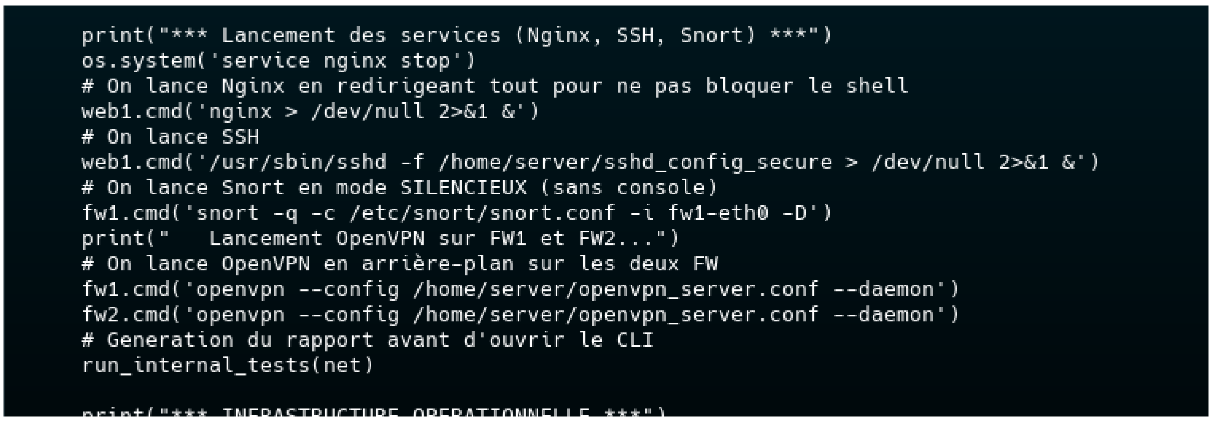
```
server@srv ~  (0.11s)
cat /home/server/admin.ovpn

client
dev tun
proto udp
remote 10.0.0.1 1194  # On vise la VIP du cluster HA
resolv-retry infinite
nobind
persist-key
persist-tun
ca /home/server/vpn-pki/pki/ca.crt
cert /home/server/vpn-pki/pki/issued/admin.crt
key /home/server/vpn-pki/pki/private/admin.key
tls-auth /home/server/vpn-pki/ta.key 1
cipher AES-256-CBC
verb 3
float
```

Étape 3 : Mise à jour du `firewall.sh`



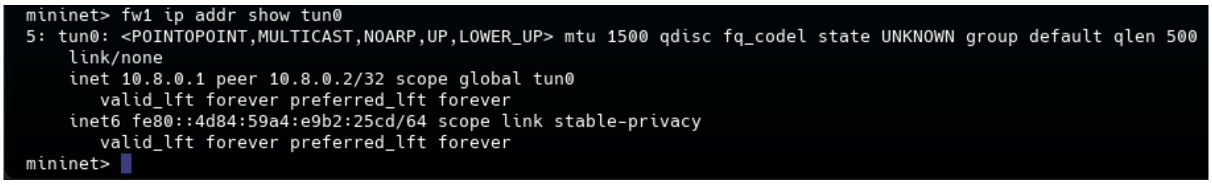
```
iptables -P OUTPUT ACCEPT

# 1. OPENVPN - Autoriser la connexion UDP 1194 depuis l'Admin (WAN)
# On autorise l'admin (10.0.0.20) à se connecter au VPN
iptables -A INPUT -p udp --dport 1194 -s 10.0.0.20 -j ACCEPT

# 2. TUNNEL - Autoriser le trafic venant du VPN (tun+) vers le LAN et la DMZ
# Les clients VPN (10.8.0.x) peuvent accéder à tout
iptables -A FORWARD -i tun+ -o fw+-eth1 -j ACCEPT  # Vers DMZ
iptables -A FORWARD -i tun+ -o fw+-eth2 -j ACCEPT  # Vers LAN

# 3. TUNNEL - Autoriser le retour
iptables -A FORWARD -i fw+-eth1 -o tun+ -j ACCEPT
iptables -A FORWARD -i fw+-eth2 -o tun+ -j ACCEPT

# 3. REGLES DE BASE (Loopback, HA, Ping)
iptables -A INPUT -i lo -j ACCEPT
iptables -A INPUT -p vrrp -j ACCEPT
iptables -A INPUT -d 224.0.0.18 -j ACCEPT
```

Étape 4 : Mise à jour du script Python (`projet_topo.py`)

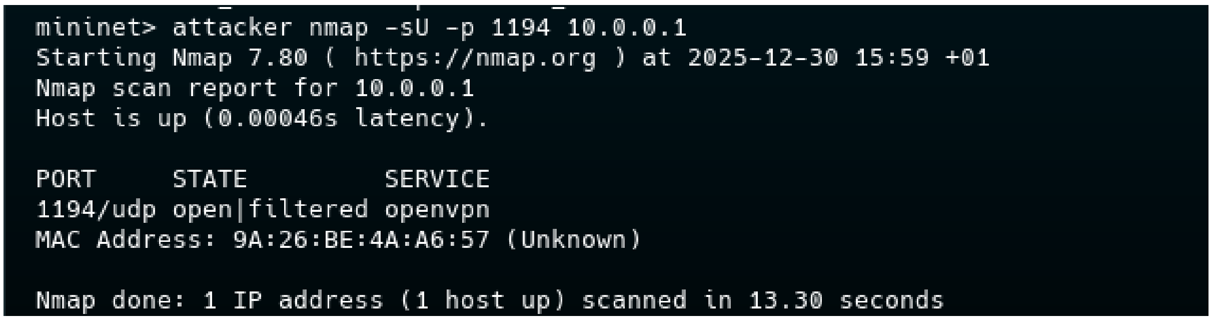


```
    print("*** Lancement des services (Nginx, SSH, Snort) ***")
    os.system('service nginx stop')
    # On lance Nginx en redirigeant tout pour ne pas bloquer le shell
    web1.cmd('nginx > /dev/null 2>&1 &')
    # On lance SSH
    web1.cmd('/usr/sbin/sshd -f /home/server/sshd_config_secure > /dev/null 2>&1 &')
    # On lance Snort en mode SILENCIEUX (sans console)
    fw1.cmd('snort -q -c /etc/snort/snort.conf -i fw1-eth0 -D')
    print("   Lancement OpenVPN sur FW1 et FW2...")
    # On lance OpenVPN en arrière-plan sur les deux FW
    fw1.cmd('openvpn --config /home/server/openvpn_server.conf --daemon')
    fw2.cmd('openvpn --config /home/server/openvpn_server.conf --daemon')
    # Generation du rapport avant d'ouvrir le CLI
    run_internal_tests(net)

    print("*** INFRASTRUCTURE OPERATIONNELLE ***")
```

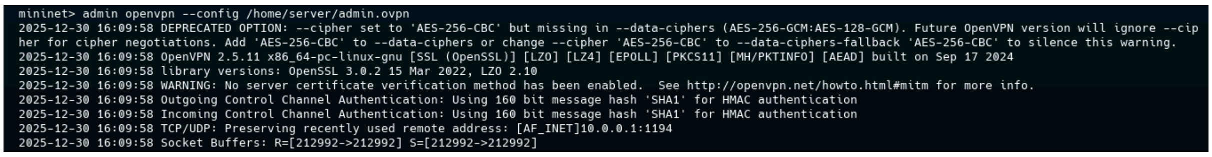
Étape 5 : Le Test Ultime (La Simulation Client)



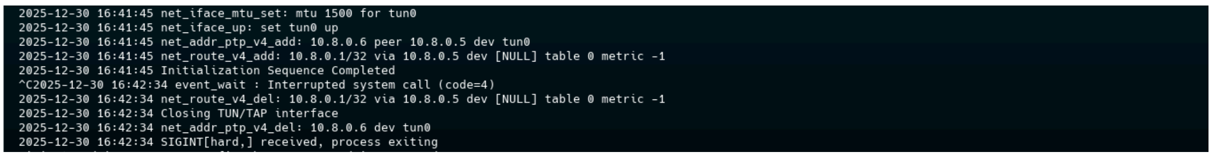
```
mininet> fw1 ip addr show tun0
5: tun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UNKNOWN group default qlen 500
    link/none
    inet 10.8.0.1 peer 10.8.0.2/32 scope global tun0
       valid_lft forever preferred_lft forever
    inet6 fe80::4d84:59a4:e9b2:25cd/64 scope link stable-privacy
       valid_lft forever preferred_lft forever
mininet>
```



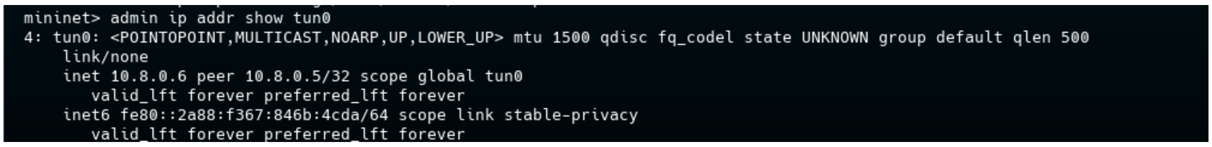
```
mininet> attacker nmap -sU -p 1194 10.0.0.1
Starting Nmap 7.80 ( https://nmap.org ) at 2025-12-30 15:59 +01
Nmap scan report for 10.0.0.1
Host is up (0.00046s latency).

PORT     STATE         SERVICE
1194/udp open|filtered openvpn
MAC Address: 9A:26:BE:4A:A6:57 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 13.30 seconds
```



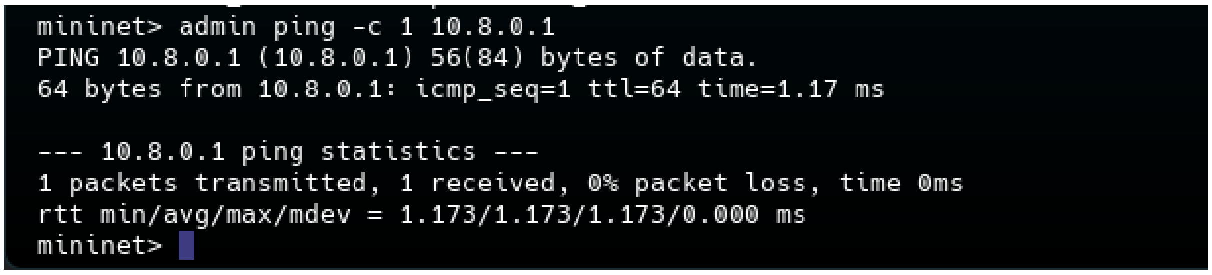
```
mininet> admin openvpn --config /home/server/admin.ovpn
2025-12-30 16:09:58 DEPRECATED OPTION: --cipher set to 'AES-256-CBC' but missing in --data-ciphers (AES-256-GCM:AES-128-GCM). Future OpenVPN version will ignore --cip
her for cipher negotiations. Add 'AES-256-CBC' to --data-ciphers or change --cipher 'AES-256-CBC' to --data-ciphers-fallback 'AES-256-CBC' to silence this warning.
2025-12-30 16:09:58 OpenVPN 2.5.11 x86_64-pc-linux-gnu [SSL (OpenSSL)] [LZO] [LZ4] [EPOLL] [PKCS11] [MH/PKTINFO] [AEAD] built on Sep 17 2024
2025-12-30 16:09:58 library versions: OpenSSL 3.0.2 15 Mar 2022, LZO 2.10
2025-12-30 16:09:58 WARNING: No server certificate verification method has been enabled.  See http://openvpn.net/howto.html#mitm for more info.
2025-12-30 16:09:58 Outgoing Control Channel Authentication: Using 160 bit message hash 'SHA1' for HMAC authentication
2025-12-30 16:09:58 Incoming Control Channel Authentication: Using 160 bit message hash 'SHA1' for HMAC authentication
2025-12-30 16:09:58 TCP/UDP: Preserving recently used remote address: [AF_INET]10.0.0.1:1194
2025-12-30 16:09:58 Socket Buffers: R=[212992->212992] S=[212992->212992]
```



```
2025-12-30 16:41:45 net_iface_mtu_set: mtu 1500 for tun0
2025-12-30 16:41:45 net_iface_up: set tun0 up
2025-12-30 16:41:45 net_addr_ptp_v4_add: 10.8.0.6 peer 10.8.0.5 dev tun0
2025-12-30 16:41:45 net_route_v4_add: 10.8.0.1/32 via 10.8.0.5 dev [NULL] table 0 metric -1
2025-12-30 16:41:45 Initialization Sequence Completed
^C2025-12-30 16:42:34 event_wait : Interrupted system call (code=4)
2025-12-30 16:42:34 net_route_v4_del: 10.8.0.1/32 via 10.8.0.5 dev [NULL] table 0 metric -1
2025-12-30 16:42:34 Closing TUN/TAP interface
2025-12-30 16:42:34 net_addr_ptp_v4_del: 10.8.0.6 dev tun0
2025-12-30 16:42:34 SIGINT[hard,] received, process exiting
```



```
mininet> admin ip addr show tun0
4: tun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UNKNOWN group default qlen 500
    link/none
    inet 10.8.0.6 peer 10.8.0.5/32 scope global tun0
       valid_lft forever preferred_lft forever
    inet6 fe80::2a88:f367:846b:4cda/64 scope link stable-privacy
       valid_lft forever preferred_lft forever
```



```
mininet> admin ping -c 1 10.8.0.1
PING 10.8.0.1 (10.8.0.1) 56(84) bytes of data.
64 bytes from 10.8.0.1: icmp_seq=1 ttl=64 time=1.17 ms

--- 10.8.0.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 1.173/1.173/1.173/0.000 ms
mininet>
```