Configuration du pare-feu avec nftables

Introduction :

Dans le cadre de la sécurisation de l'infrastructure réseau, la configuration d'un pare-feu constitue une étape essentielle. Ce script met en œuvre **nftables**, le successeur moderne **d'iptables**, afin de contrôler et filtrer le trafic circulant entre les différentes zones réseau.

L'objectif de cette configuration est double :

- **Filtrer finement les flux interzones**, en autorisant uniquement les communications nécessaires entre les sous-réseaux (**LAN**, **DMZ**, **WiFi**, **ToIP**, **VPN**, **etc**.) ;
- **Assurer la translation d'adresses (NAT)** vers Internet via l'interface externe (**eth0**), afin de masquer les adresses privées des clients internes.

Le pare-feu repose sur une politique par défaut restrictive (**drop**) pour les chaînes **input** et **forward**, garantissant que seul le trafic explicitement autorisé est accepté. Les règles tiennent compte des protocoles applicatifs (**HTTP**, **HTTPS**, **DNS**, **RADIUS**, etc.), des interfaces réseau, ainsi que des états de connexion (**ct state established**,**related**) pour limiter les ouvertures injustifiées.

Cette approche assure une **séparation stricte des zones de sécurité**, tout en maintenant la fonctionnalité nécessaire au bon fonctionnement des services et à la communication entre les machines autorisées.

1. Mise à jour du système

```
#!/bin/bash

# Met à jour les paquets
echo "[+] Mise à jour du système..."
apt update && apt upgrade -y
```

- Met à jour la liste des paquets disponibles
- Installe les mises à jour de sécurité et fonctionnalités

2. Installation des paquets nécessaires

```
# Installe les paquets nécessaires
echo "[+] Installation de nftables, nano, frr..."
apt install -y nftables nano frr
```

- **nftables** : nouveau framework de filtrage réseau (remplaçant d'iptables)

- **nano** : éditeur de texte simple
- **frr** : suite de routage (Free Range Routing) pour les protocoles de routage dynamique

3. Configuration nftables - Structure générale

Le script crée `/etc/nftables.conf` avec deux tables principales :

Table FILTER (sécurité)

Gère le filtrage des paquets avec 3 chaînes :

```
chain input {
    type filter hook input priority 0;
    policy drop;

    ct state established,related accept
    iif lo accept

    # SSH et ICMP en local
    ip protocol icmp accept
    tcp dport 22 accept

    # RADIUS
    udp dport {1812,1813} accept

    # DNS pour DMZ
    iifname "eth0" udp dport 53 accept
    iifname "eth0" tcp dport 53 accept

    # HTTP/HTTPS pour DMZ
    iifname "eth0" tcp dport {80,443} accept

    # Proxy (squid)
    tcp dport 3128 accept
```

Chain INPUT (trafic entrant vers le firewall)

- **Politique par défaut** : DROP (tout bloquer)
- **Connexions établies** : autorisées (ct state established,related)
- **Interface loopback** : autorisée (trafic local)
- **Services autorisés** :
    - ICMP (ping)
    - SSH (port 22)
    - RADIUS (ports 1812, 1813) pour l'authentification
    - DNS (ports 53) depuis eth0 pour la DMZ
    - HTTP/HTTPS (ports 80, 443) depuis eth0 pour la DMZ
    - Proxy Squid (port 3128)

```
chain forward {
    type filter hook forward priority 0;
    policy drop;

    ct state established,related accept

    # LAN ↔ DMZ
    ip saddr 192.168.10.0/24 ip daddr 192.168.100.0/24 accept
    ip saddr 192.168.100.0/24 ip daddr 192.168.10.0/24 accept

    # Client 12.12.12.0/24 → DMZ (HTTP/HTTPS)
    ip saddr 12.12.12.0/24 ip daddr 192.168.100.0/24 tcp dport {80,443} accept

    # VPN → LAN
    ip saddr 10.10.10.0/24 ip daddr 192.168.10.0/24 accept

    # LAN → Internet
    ip saddr 192.168.10.0/24 oifname "eth0" accept

    # ToIP ↔ LAN
    ip saddr 192.168.30.0/24 ip daddr 192.168.10.0/24 accept
    ip saddr 192.168.10.0/24 ip daddr 192.168.30.0/24 accept

    # ToIP → Auth
    ip saddr 192.168.30.0/24 ip daddr 192.168.200.0/24 udp dport 1812 accept

    # WiFi → Internet
    ip saddr 192.168.20.0/24 oifname "eth0" accept
```

```
    # WiFi → Auth (RADIUS)
    ip saddr 192.168.20.0/24 ip daddr 192.168.200.0/24 udp dport 1812 accept

    # Invités → Proxy
    ip saddr 192.168.50.0/24 ip daddr 192.168.100.0/24 tcp dport 3128 accept

    # Auth ↔ LAN/WiFi/ToIP
    ip saddr 192.168.10.0/24 ip daddr 192.168.200.0/24 accept
    ip saddr 192.168.20.0/24 ip daddr 192.168.200.0/24 accept
    ip saddr 192.168.30.0/24 ip daddr 192.168.200.0/24 accept

    ip saddr 192.168.200.0/24 ip daddr 192.168.10.0/24 accept
    ip saddr 192.168.200.0/24 ip daddr 192.168.20.0/24 accept
    ip saddr 192.168.200.0/24 ip daddr 192.168.30.0/24 accept
}
```

Chain FORWARD (trafic traversant le firewall)

- **Politique par défaut** : DROP
- **Règles inter-zones définies** :

LAN ↔ DMZ (192.168.10.0/24 ↔ 192.168.100.0/24)

- Autorise communication LAN-DMZ

Clients externes → DMZ (12.12.12.0/24 → 192.168.100.0/24)

- Accès HTTP/HTTPS uniquement

VPN → LAN (10.10.10.0/24 → 192.168.10.0/24)

- Accès depuis le VPN vers le réseau local

LAN → Internet

- Sortie Internet autorisée pour le LAN

ToIP ↔ LAN (192.168.30.0/24 ↔ 192.168.10.0/24)

- Communication bidirectionnelle (téléphonie IP)

ToIP → Authentification (192.168.30.0/24 → 192.168.200.0/24)

- RADIUS pour l'auth des téléphones

WiFi → Internet/Auth (192.168.20.0/24)

- Sortie Internet et accès RADIUS

Invités → Proxy (192.168.50.0/24 → 192.168.100.0/24)

- Accès Internet via proxy uniquement

Serveur Auth ↔ Réseaux (192.168.200.0/24)

- Communication bidirectionnelle avec LAN, WiFi, ToIP

Chain OUTPUT (trafic sortant du firewall)

```
    chain output {
        type filter hook output priority 0;
        policy accept;
    }
```

- **Politique** : ACCEPT (tout autorisé)

Table NAT (traduction d'adresses)

```
table ip nat {
    chain postrouting {
        type nat hook postrouting priority 100;
        oifname "eth0" masquerade
    }
}
```

- **Masquerading** : traduit les adresses privées en adresse publique lors de la sortie par eth0

4. Activation et vérification

```
# Démarre nftables
echo "[+] Démarrage de nftables avec la nouvelle configuration..."
nft -f /etc/nftables.conf

# Vérifie que les règles sont bien appliquées
echo "[+] Règles nftables actuellement en place :"
nft list ruleset
```

- Charge la configuration
- Affiche les règles pour vérification

5. Activation de l'IP Forwarding

```
# Active l'IP forwarding
echo "[+] Activation de l'IP forwarding..."
sysctl -w net.ipv4.ip_forward=1
grep -q '^net.ipv4.ip_forward=1' /etc/sysctl.conf || echo "net.ipv4.ip_forward=1" >> /
```

- Active le routage IP (nécessaire pour que le firewall route les paquets)
- Rend le paramètre persistant au redémarrage

6. Configuration FRR (Routage RIP)

Activation du démon RIP

```
# --- Configuration de FRR ---
echo "[+] Configuration de FRR..."

# Active le démon RIP uniquement
cat <<EOF > /etc/frr/daemons
bgpd=no
ospfd=no
ospf6d=no
ripd=yes
ripngd=no
EOF
```

Configuration RIP

```
# Crée la configuration RIP
cat <<EOF > /etc/frr/frr.conf
frr defaults traditional
hostname FIREWALL
log file /var/log/frr/frr.log

router rip
 version 2
 network 192.168.10.0/24
 network 192.168.20.0/24
 network 192.168.30.0/24
 network 192.168.50.0/24
 network 192.168.100.0/24
 network 192.168.200.0/24
 network 11.11.11.0/24
EOF
```

- **RIP v2** : protocole de routage dynamique
- **Réseaux annoncés** : tous les segments réseau gérés par le firewall
- Permet aux autres routeurs du réseau de connaître les routes

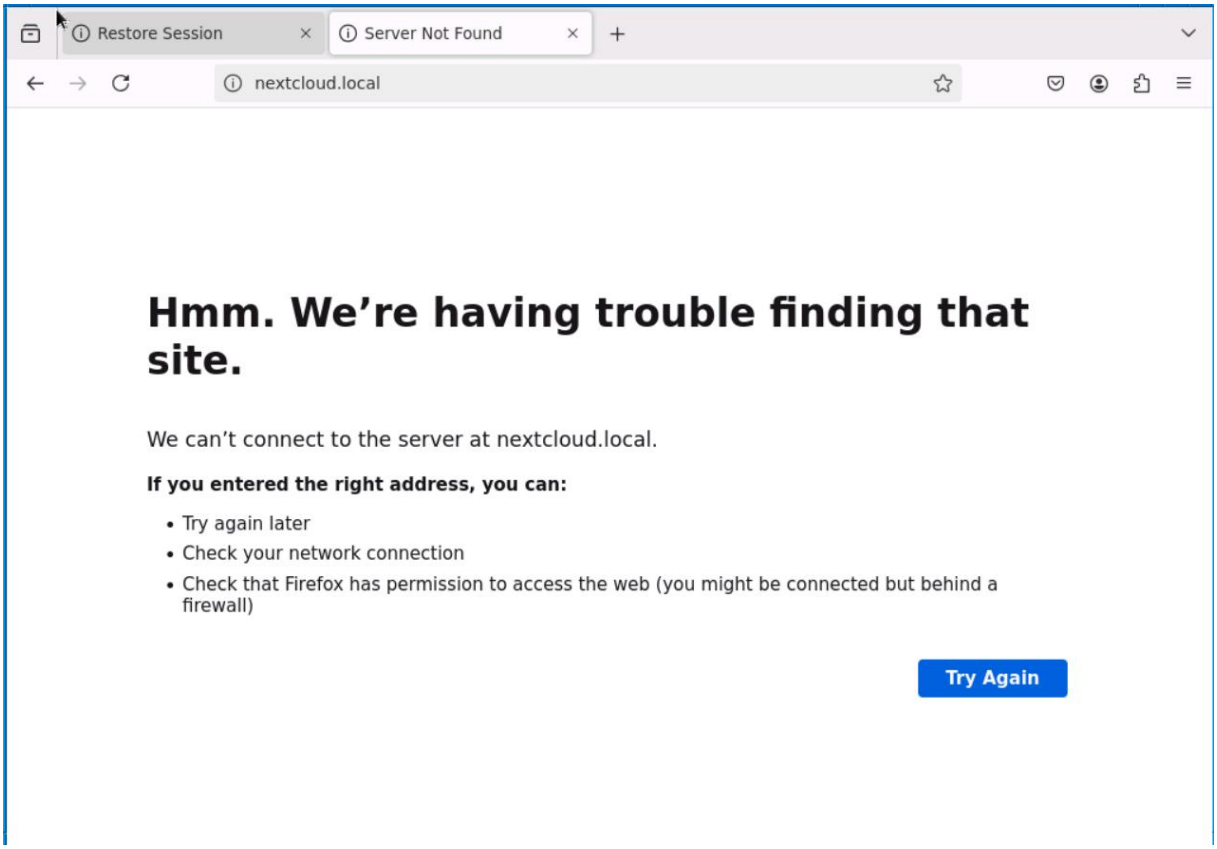
7. Redémarrage des services

```
# Redémarre manuellement les daemons FRR
echo "[+] Redémarrage manuel de FRR..."
/usr/lib/frr/frrinit.sh stop
sleep 2
/usr/lib/frr/frrinit.sh start

echo "[✓] Configuration terminée : nftables, IP forwarding et FRR (RIP) sont opération
```

- Redémarre proprement les démons FRR pour appliquer la configuration

Test d'acces :



Nous voyons voir qu'un client externe n'as pas accés a notre service web interne.

On fait le test http://smartech.site dans le navigateur du webterm invité (192.168.50.11) d' un site web qu'on a cree dans le dmz .

Nous voyons ici qu'un client externe a acces a notre service web.

Conclusion :

La configuration réalisée à travers ce script met en place un **pare-feu centralisé et structuré** à l'aide de **nftables**, couplé à une gestion du routage dynamique via le protocole **RIP** (avec **FRR**). Elle permet de **maîtriser les flux réseau entre les différentes zones** (**LAN**, **DMZ**, **WiFi**,

ToIP, **VPN**, **invités**…) tout en **assurant une sortie sécurisée vers Internet** grâce au mécanisme de **NAT**.

Chaque règle a été définie pour **répondre à un besoin fonctionnel précis**, sans compromis sur la sécurité. Le trafic non autorisé est bloqué par défaut, garantissant une **politique de sécurité stricte** et efficace. L'activation du **forwarding IP**, combinée à la redondance dynamique apportée par **RIP**, assure également une **connectivité interzones fiable et évolutive**.

En résumé, ce pare-feu offre un **socle robuste et sécurisé** pour un réseau d'entreprise multi-zones, tout en restant adaptable à de futurs besoins ou extensions.