Sécurisation du réseau et défense contre les attaques DDoS à l'aide de pare-feu sous Linux

Partie 0 : Comparatif des solutions de pare-feu:

1) iptables :

Iptables est le pare-feu traditionnel pour Linux, utilisé depuis des années.
Il fonctionne en utilisant des tables et des chaînes pour définir les règles de filtrage du trafic réseau.
Il permet de surveiller et filtrer le trafic entrant et sortant selon des règles définies par l'utilisateur, renforçant ainsi la sécurité du serveur.
Avec iptables, les utilisateurs peuvent autoriser sélectivement le trafic sur leur système en configurant des règles spécifiques.
Il fonctionne en utilisant des tableaux de filtrage de paquets, avec chaque tableau contenant plusieurs chaînes de règles.
Chaque règle dans une chaîne spécifie une action à effectuer sur les paquets correspondants, leur attribuant une cible particulière.
Iptables reste une option puissante et flexible pour sécuriser les systèmes Linux.

nftables :

nftables est une alternative à iptables conçue pour résoudre ses limitations, notamment en termes de performance lors de la gestion de règles volumineuses.
Il offre une syntaxe plus simple et expressive, permettant de réduire le nombre de règles nécessaires pour une efficacité équivalente.
Avec nftables, les utilisateurs bénéficient d'une gestion efficace des règles de pare-feu et de performances améliorées.
Cette solution, intégrée au noyau Linux, est plus moderne et flexible que iptables, offrant une totale liberté dans le nommage des tables et des chaînes et permettant d'éviter des traitements inutiles.
C'est un bon choix pour ceux qui recherchent une solution plus moderne et plus flexible que iptables.

firewalld :

Firewalld est un pare-feu dynamique pour Linux offrant une gestion simplifiée via des zones et des services.
Il propose une interface en ligne de commande et une interface graphique conviviale pour configurer les règles de pare-feu.
Firewalld permet une gestion flexible des règles en temps réel, s'appuyant sur nftables ou iptables selon la configuration du système.
Il convient aux environnements où les besoins de sécurité évoluent fréquemment, offrant une mise en œuvre et une paramétrisation faciles par rapport à iptables ou nftables.

Il convient particulièrement aux environnements où les besoins de sécurité évoluent fréquemment.

UFW (Uncomplicated Firewall) :

UFW est une interface simplifiée pour iptables, adaptée aux utilisateurs débutants, offrant une syntaxe simplifiée et des profils pré-définis pour configurer facilement le pare-feu.
Bien que pratique pour une configuration rapide, il peut manquer de fonctionnalités avancées présentes dans d'autres solutions telles que iptables, nftables ou Firewalld.
Le choix entre ces pare-feu dépendra des besoins spécifiques de l'utilisateur en matière de sécurité et de la complexité qu'il est prêt à gérer.

2) Caractéristiques principales :

Iptables utilise une syntaxe relativement complexe pour configurer les règles de pare-feu.
Les règles sont organisées en tables et chaînes, avec des cibles spécifiques pour chaque règle.
Iptables peut filtrer le trafic IPv4 et IPv6.

Nftables offre une syntaxe plus simple et plus expressive que iptables.
Il permet une gestion plus efficace des règles de pare-feu avec des performances améliorées.
Il prend en charge à la fois IPv4 et IPv6 et propose une approche plus flexible pour la configuration des règles de pare-feu.

Similitudes :

Les deux sont des solutions de pare-feu pour les systèmes Linux.
Ils permettent de filtrer le trafic entrant et sortant en fonction des règles définies par l'utilisateur.
Ils utilisent des tables, des chaînes et des cibles pour organiser les règles de pare-feu.
Les deux prennent en charge IPv4 et IPv6.

Différences :

La syntaxe, iptables utilise une syntaxe plus complexe, alors que nftables propose une syntaxe plus simple et expressive, ce qui facilite la configuration des règles de pare-feu pour les utilisateurs.
Nftables est conçu pour offrir des performances améliorées par rapport à iptables.
Nftables offre une gestion plus flexible des règles de pare-feu, ce qui lui permet de mieux répondre aux besoins évolutifs en matière de sécurité.
Nftables est considéré comme le successeur à long terme d'iptables et est intégré dans le noyau Linux, ce qui en fait un choix plus moderne.

3)

	Avantage	Inconvénient
Iptables	-Syntaxe familière pour les utilisateurs existants. -Documentation abondante. -Intégré depuis longtemps dans les distributions Linux.	-Syntaxe complexe et moins intuitive pour les débutants. -Performances moins bonnes lors de la mise à jour des règles en grand nombre. -Gestion moins flexible des règles de pare-feu. -Fragmentation des outils avec iptables, ip6tables, arptables et ebtables.
Nftables	-Syntaxe simplifiée et plus expressive. -Performances améliorées lors de la mise à jour des règles. -Gestion flexible des règles de pare-feu. -Solution centralisée pour IPv4, IPv6 et Ethernet.	-Nécessite un apprentissage pour les utilisateurs habitués à iptables. -Adoption plus récente et documentation moins abondante que iptables. -Certaines fonctionnalités avancées d'iptables peuvent manquer.

Étapes des commandes:

MAJ de la VM:

su root
mdp root
sudo apt update && sudo apt upgrade -y

Installation de python et pip et ssh:

sudo apt install python3
sudo apt install python3-pip
sudo pip install flask
sudo apt install openssh-server
sudo systemctl enable ssh
sudo systemctl start ssh

Création de l'espace de travail:

```
cd ~
mkdir python
cd python
nano app80.py
```
code de flask pour le port 80
```
nano app5001.py
```
code de flask pour le port 5001
```
sudo python3 app80.py
```

Mise en place du pare-feu:

```
sudo iptables -A INPUT -i lo -j ACCEPT
sudo iptables -A INPUT -p tcp –dport 80 -j ACCEPT
sudo iptables -A INPUT -p tcp –dport 443 -j ACCEPT
sudo iptables -A INPUT -p tcp –dport 22 -j ACCEPT
sudo iptables -I INPUT -p tcp –syn –dport 80 -m connlimit –connlimit-above 10
–connlimit-mask 32 -j REJECT –reject-with tcp-reset

sudo iptables -A OUTPUT -o lo -j ACCEPT
sudo iptables -A OUTPUT -d 51.12.246.68 -j ACCEPT

sudo iptables -P INPUT DROP
sudo iptables -P FORWARD DROP
sudo iptables -P OUTPUT DROP

sudo iptables-save > backup_iptables.txt
```

Hors script :

code de flask pour le port 80 :

```
from flask import Flask
app = Flask(__name__)
@app.route('/')
def home():
        return "Hello, world!<br/><br/> Vous êtes bien sur mon serveur Flask"

if __name__=="__main__":
        app.run(debug=True, hots="0.0.0.0", port=80)
```

code de flask pour le port 5001 :

```
from flask import Flask
app = Flask(__name__)
```

```
@app.route('/')
def home():
        return "Hello, world!<br/><br/> Vous êtes bien sur mon serveur Flask"

if __name__=="__main__":
        app.run(debug=True, hots="0.0.0.0", port=5001)
```

pour kill un script python qui tourne pour lequel on n'a plus accès au terminal:
sudo pkill -f script_python.py

Partie 2 : Gestion des attaques DDoS

Installation d'ApacheBench :

sudo apt install apache2-utils

Utilisation d'ApacheBench pour simuler une charge de requêtes sur le serveur web :

ab -n 10000 -c 1000 http://example.com/ -> ici on remplace l'url par celui du serveur web qui est :

ab -n 10000 -c 1000 http://172.20.10.6/
-n 10000: signifie le nombre total de requêtes à envoyer au serveur.
-c 1000: Cela spécifie le nombre de requêtes simultanées à envoyer au serveur.

Créer une nouvelle règle iptables: rejeter automatiquement les requêtes si un nombre excessif de connexions entrantes provient de la même adresse IP:

sudo iptables -A INPUT -p tcp --syn --dport 80 -m connlimit --connlimit-above 10
--connlimit-mask 32 -j REJECT --reject-with tcp-reset

-m connlimit --connlimit-above 10 : limite le nombre de connexions à 10.
--connlimit-mask 32 : applique la limite à chaque adresse IP.
-j REJECT --reject-with tcp-reset : rejette la connexion avec un message TCP RESET, ce qui signifie que la connexion sera fermée brutalement.

Le nombre excessif de connexions entrantes provenant de la même adresse IP dépendra de plusieurs facteurs, notamment la capacité de traitement de la machine virtuelle. Ici on l'a limité à 10.

Partie 3 : Amélioration de la défense

1)

- À l'heure actuelle, la configuration se focalise que sur la limitation du nombre de connexions par adresse IP dans une période spécifique.

- Attaques DDoS distribuées : Les attaques DDoS distribuées perturbent cette configuration en utilisant plusieurs sources d'attaques provenant de diverses adresses IP.

- Les attaques DDoS avancées: Il est envisageable que les attaques DDoS soient complexes et optent pour différentes méthodes afin de contourner les mesures de sécurité simples, ce qui requiert des solutions de protection plus performantes.

2)

- Utilisation de services de mitigation DDoS spécialisés :

  Les services de mitigation DDoS (pratique consistant à supprimer ou réduire l'impact d'attaques par déni de service) peuvent offrir une protection avancée contre les attaques DDoS en utilisant des techniques telles que la surveillance du trafic en temps réel, la filtration intelligente du trafic malveillant et la répartition de la charge sur plusieurs centres de données pour atténuer l'impact des attaques.

- Utilisation de répartiteurs de charge intelligents :

  Les répartiteurs de charge intelligents peuvent identifier et filtrer le trafic malveillant provenant d'attaques DDoS, tout en dirigeant le trafic légitime vers le serveur.
  Ils peuvent également distribuer efficacement la charge du trafic entrant sur plusieurs serveurs pour répartir la charge et maintenir les services en ligne en cas d'attaque.

- Mise en place de pare-feu d'application (WAF) :

  Les pare-feu d'application web (WAF) peuvent détecter et bloquer les attaques de couche applicative telles que les attaques par injection SQL, les attaques par déni de service HTTP (HTTP DoS), etc.
  Ils peuvent également aider à atténuer les effets des attaques DDoS en filtrant le trafic malveillant au niveau de l'application.

- Utilisation de systèmes de détection d'intrusion (IDS) et de prévention d'intrusion (IPS) :

  Les systèmes de détection d'intrusion (IDS) et de prévention d'intrusion (IPS) peuvent surveiller le trafic réseau et détecter les schémas de trafic malveillant associés aux attaques DDoS.
  Ils peuvent automatiquement bloquer le trafic suspect ou alerter les administrateurs pour une action appropriée.

- Mise en place de CDN (Content Delivery Network) :

  Les CDN peuvent aider à atténuer les attaques DDoS en distribuant le contenu statique sur des serveurs répartis géographiquement.
  Cela peut réduire la charge sur le serveur d'origine et offrir une capacité supplémentaire pour absorber le trafic lors d'une attaque DDoS.

- Utilisation de solutions de protection basées sur l'IA et le machine learning :

  Les solutions basées sur l'intelligence artificielle (IA) et l'apprentissage automatique (machine learning) peuvent analyser le trafic en temps réel et détecter les anomalies ou les comportements suspects associés aux attaques DDoS.
  Ils peuvent s'adapter et évoluer pour mieux défendre contre les attaques DDoS de plus en plus sophistiquées.