Pare feu Linux

Configurer un pare feu sous Linux avec NetFilter/IPTables

Sommaire

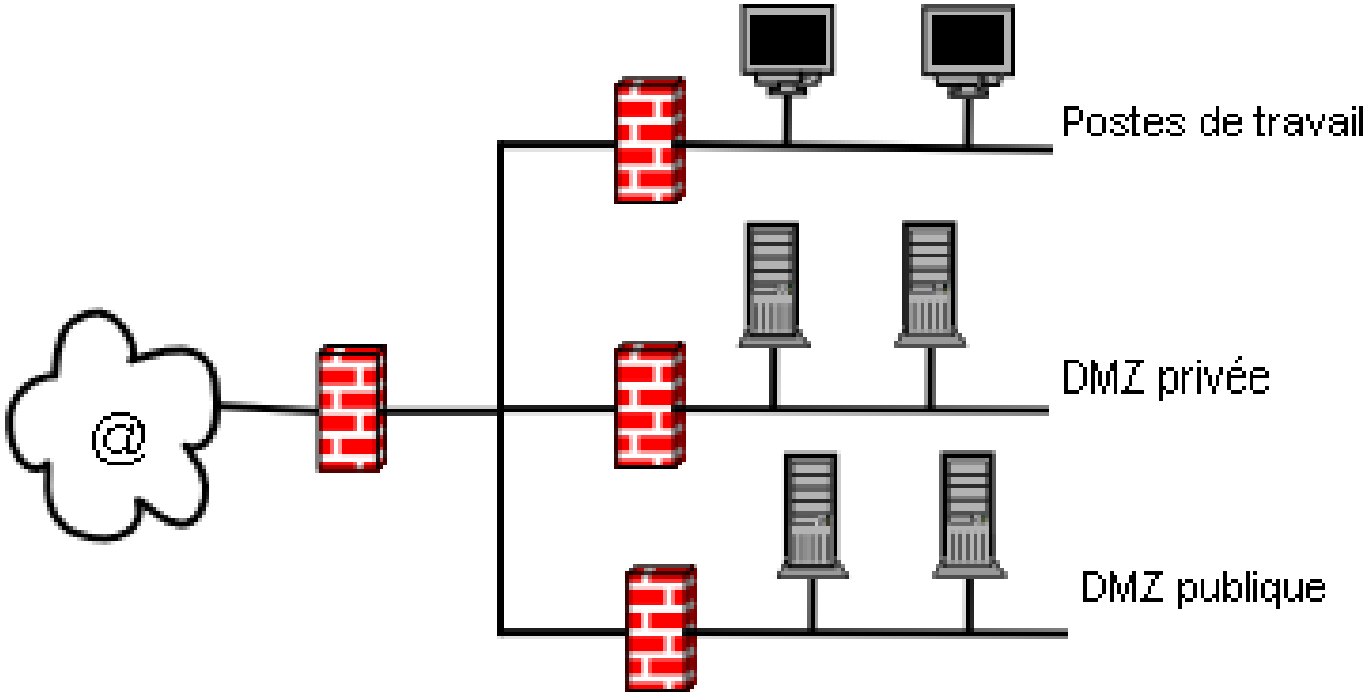
- Introduction
- Principes généraux et bonnes pratiques
- Présentation de NetFilter/IPTables
- Prise en main
  - Syntaxe
  - Filtrage simple
  - NAT
  - Redirection
- Mise en pratique

Introduction

- Netfilter
  - API TCP/IP pour noyau Linux
  - Permet à des programmes d'agir sur la pile TCP/IP
  - Parmi ces programmes : IPTables
- IPTables
  - Utilise NetFilter pour apporter à une machine Linux les fonctionnalités nécessaires au filtrage réseau TCP/IP:
    - Filtrage niveau 2 : adresses MAC
    - Filtrage niveau 3 : adresses IP
    - Filtrage niveau 4 : UDP/TCP/ICMP
    - Filtrage niveau 7 en utilisation des extensions Netfilter

Principes généraux

- Principes d'architecture
  - Création de zones pour isoler les machines les plus sensibles



Principes généraux

- Installation de l'OS (Ex. Debian)
  - Choisir une distribution stable
  - Installation minimale
  - Nettoyage
    - Arrêt des services inutiles
    - Désinstallation des logiciels inutiles
    - Suppression des comptes systèmes inutiles
    - Création de comptes non root
    - Mise à jour et application des derniers correctifs
    - Optionnellement : recompilation du noyau

Principes généraux

- Un pare feu est un routeur filtrant :
  - Plusieurs interfaces réseaux, 1 par zone + 1 pour l'administration du pare feu
  - Activation du routage
    - echo « 1 » > /proc/sys/net/ipv4/ip_forward
  - Choix de la route par défaut et configuration du routage vers les autres zones
  - Dans le cas d'un pare feu NetFilter/IPTables, les règles sont positionnées en FORWARD

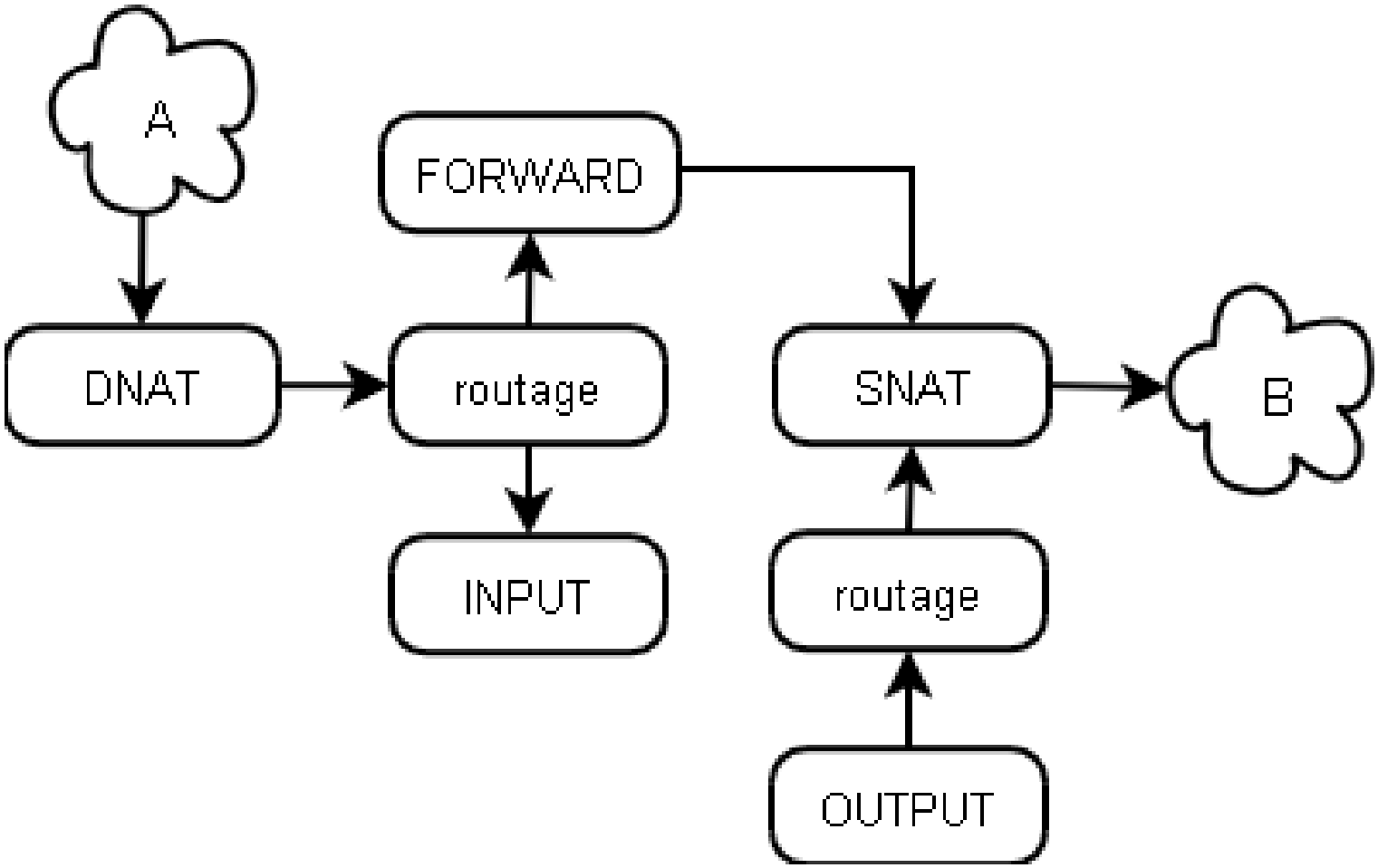
Principes généraux

- Sécuriser la machine et son OS
  - Paramètres sysctl à positionner pour :
    - contrer certaines attaques
    - limiter la vulnérabilité de la pile TCP/IP
  - Administration de la machine
    - Distante
      - OpenSSH pour connexion et transferts de fichiers
    - Locale
      - Uniquement depuis la console
      - Ne pas oublier la sécurité physique
      - Baie fermée, salle à accès contrôlé.

NetFilter/IPTables

- API pour la pile TCP/IP du noyau Linux
- Fournit des points d'entrée dans cette pile qui permettent à des programmes d'agir sur le cheminement et le traitement des paquets.
- IPTables utilise ces points d'entrée
  - Filtrage
  - Modification des caractéristiques : NAT par exemple
- NetFilter est également utilisé par des applications de gestion de trafic (Trafic Shaping)

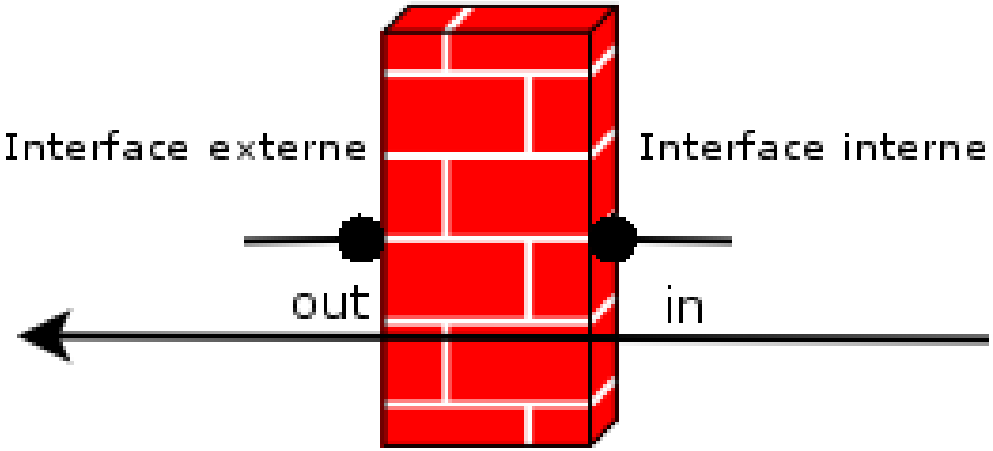
Principes NetFilter



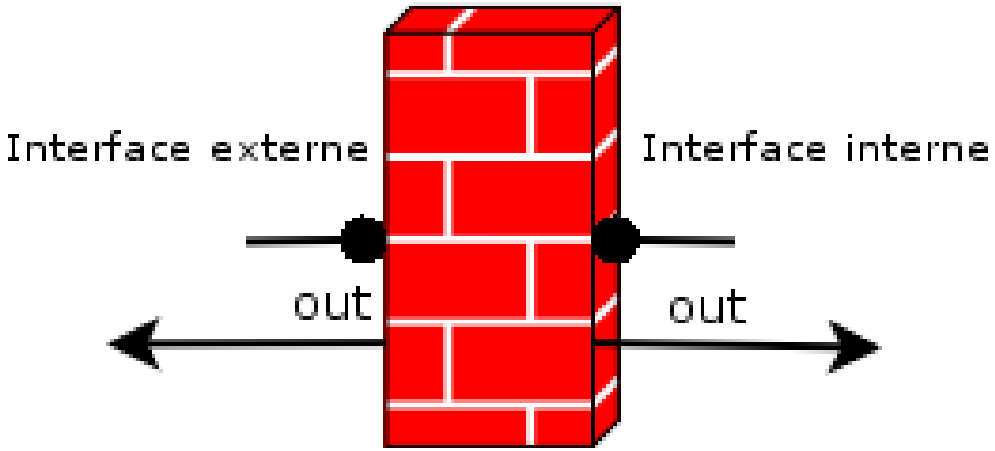
Principes Netfilter

- Réception d'un paquet par la machine
  - DNAT : modification IP destination (optionellement)
  - Routage : le paquet est-il destiné à la machine ?
    - Oui : INPUT
    - Non
      - Choix de la route vers la destination
      - Si la route existe : FORWARD
      - SNAT : modification IP source (optionellement)
- Emission d'un paquet par la machine
  - Routage
    - Choix de la route vers la destination
    - Si la route existe : OUTPUT
    - SNAT : modification IP source (optionellement)

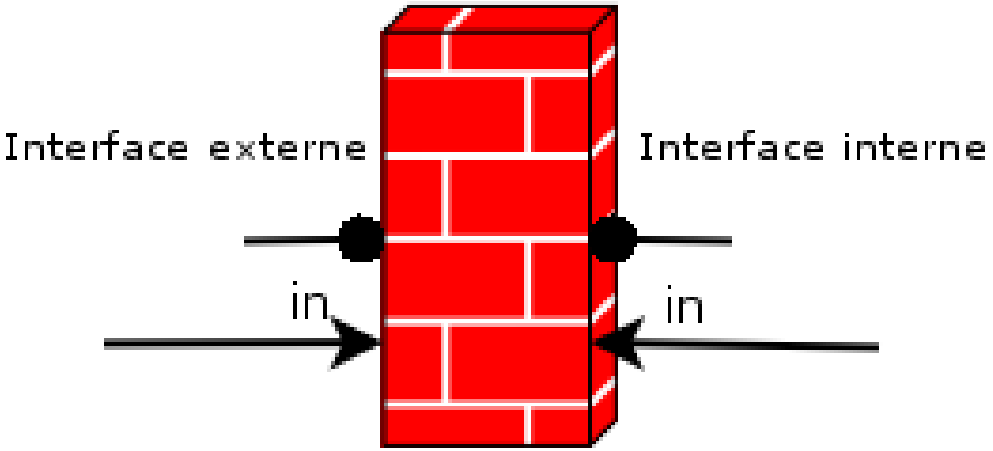
Note sur la direction (1)



Note sur la direction (2)



Note sur la direction (3)



IPTables

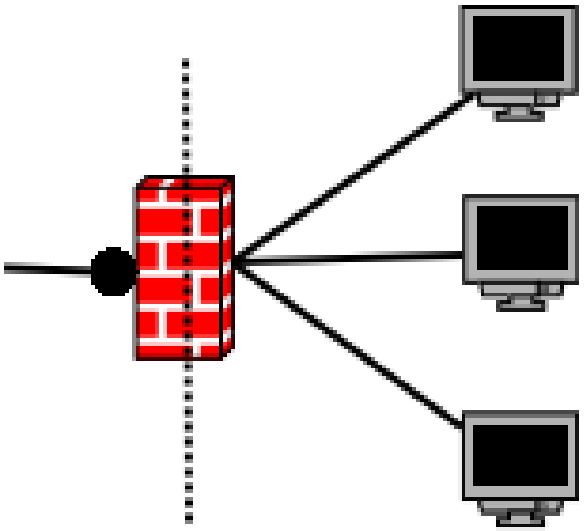
- IPTables
  - Fournit des utilitaires pour manipuler les paquets
    - iptables
- Utilise des tables (d'où sont nom !)
  - MANGLE
    - Marquage des paquets, modification des TTL et TOS
    - Utilisée pour Trafic Shaping et QOS
  - NAT
    - Translation d'adresses : SNAT, DNAT, MASQUERADE
  - FILTER
    - Filtrage : ACCEPT ou DROP

Tables / Chaines / Cibles

Table	*Chaine*	*Cible*	*Description*
MANGLE	PREROUTING, POSTROUTING	MARK, TOS, TTL	MARK : marquage du paquet, aucune modification TOS : modification du champ Type Of Service TTL : modification du TTL
NAT	PREROUTING	DNAT	Modification IP destination
	POSTROUTING	SNAT, MASQUERADE	Modification IP source
FILTER	INPUT	ACCEPT, DROP, REJECT	DROP : rejet sans avertissement REJECT : rejet avec avertissement (Ex. : TCP RST, message ICMP)
	OUTPUT		
	FORWARD		

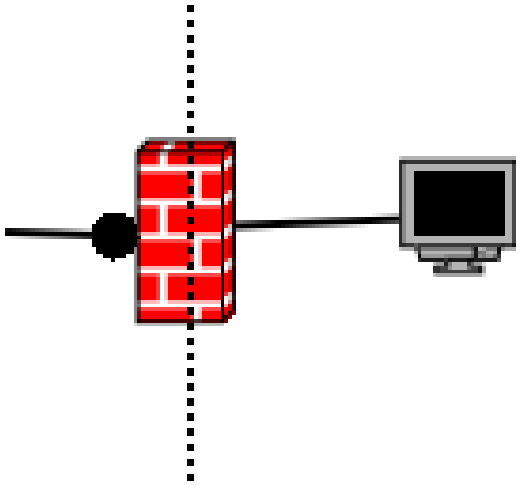
Translation d’adresses

- Masquerade
  - Plusieurs adresses sont translatées en une seule.



Translation d’adresses

- NAT 1 pour 1
  - 1 adresse et une seule est translatée en une autre.
  - Table NAT : SNAT et DNAT



Redirection

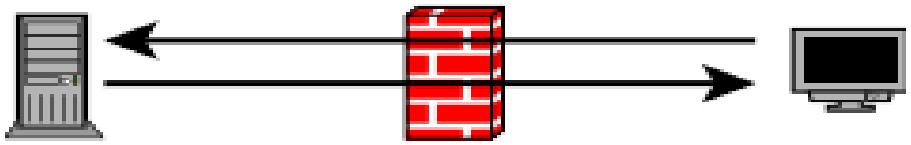
- C’est une forme particulière de NAT qui permet de modifier dynamiquement non seulement les adresses source et destination, mais aussi les ports.
- Utile pour forcer le passage d’un type de flux par un proxy
  - Exemple le plus courant : redirection des flux HTTP vers un proxy Squid

Stateful inspection (1)

- Permet de spécifier l'état d'un paquet dans une session.
- Facilite l'écriture des règles, augmente leur pertinence et améliore les performances du pare feu : si un paquet appartient à une session autorisée établie, il est transmis sans passage par les règles.
  - On utilise une table des sessions en cours. Le contenu de cette table est utilisé plutôt que les règles pour déterminer l'action qu'il faut appliquer à un paquet.

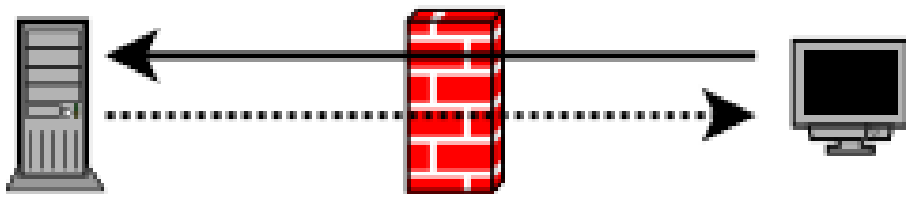
Stateful inspection (2)

- Sans Stateful il faut écrire une règle pour chaque flux :
  - Exemple d'une connexion HTTP :
    - Autorisation du premier paquet TCP :
      - iptables -A FORWARD -p tcp -s 192.168.1.0/32 –sport 1024: --syn -d any –dport 80 -j ACCEPT
    - Autres paquets de la session :
      - iptables -A FORWARD -p tcp -s 192.168.1.0/32 –sport 1024: -d any –dport 80 -j ACCEPT
    - Réponses du serveur :
      - iptables -A FORWARD -p tcp -s any –sport 80 -d 192.168.1.0/32 –dport 1024: -j ACCEPT



Stateful inspection (3)

- Avec la fonctionnalité Stateful, il suffit d'écrire la règle pour le sens « Aller », Netfilter déduit la règle de retour automatiquement.
- Dans la pratique :
  - iptables -A FORWARD -s 192.168.1.0/32 -d any -m state –state NEW -j ACCEPT
  - iptables -A FORWARD -m state –state ESTABLISHED,RELATED -j ACCEPT



Stateful inspection (4)

- Cas du protocole UDP
  - Protocole sans connexion, pas de notion de début ni de fin de session contrairement à TCP.
  - NetFilter maintient une table pour les paquets UDP. Cette table contient l'historique des sessions UDP en cours.
  - NetFilter utilise aussi un timeout : dès qu'une session UDP référencée dans la table atteint ce timeout, la session est considérée comme terminée.
- La même méthode est utilisée pour le protocole ICMP.

Stateful Instpection (5)

- Iptables reconnaît 4 états pour un paquets :
  - NEW
    - Nouvelleconnexion
    - Ouverture de session TCP par exemple
  - ESTABLISHED
    - Appartient à une connexion
  - RELATED
    - En relation avec une connexion établie
      - Exemple : messages d'erreur ICMP
  - INVALID
    - Etat non identifié. Généralement associéà DROP

Extensions NetFilter

- Installées sous forme de modules
- Permettent de traiter des protocoles complexes
  - FTP, ICQ, IRC

Chaines

- IPTables utilise des chaines (héritage du pare feu ipchains pour noyaux Linux de versions > 2)
- Chaines prédéfinies
  - PREROUTING
  - INPUT
  - FORWARD
  - OUTPUT
  - POSTROUTING
- Il est possible de définir des chaines supplémentaires
  - Elles seront toujours rattachées à une chaine prédéfinie

Cibles

- Utilisées au sein des chaines pour définir les actions
  - ACCEPT : comme son nom l'indique, laisse passer un paquet
  - REJECT : comme DROP mais informe la source (Messages ICMP)
  - DROP : rejette un paquet
  - LOG
  - DNAT : translation 1 pour 1, change l'adresse de destination
  - SNAT : translation 1 pour 1, change l'adresse de source
  - MASQUERADE : translation 1 pour N
  - REDIRECT : modifie adresse et ports

Syntaxe

- Format d'une règle
  - iptables -t table command match -j target
- Table
  - -t NAT, -t MANGLE, etc
  - Par défaut : -t FILTER
- Command
  - -A : ajoute la règle en fin de liste
  - -D : efface une règle
  - -I : insère la règle en début de liste
  - -F : flush les règles
  - -P : politique par défaut
    - iptables -P FORWARD DROP

Syntaxe

- Match
  - Définit les critères des paquets
  - Types génériques :
  - -p : protocole. TCP/UDP/ICMP
    - -p tcp
  - -s : adresse IP source
    - Littérale : -s 192.168.1.1
    - Réseau : -s 192.168.1.0/32
  - -d : adresse IP destination
  - -i : interface physique d'entrée
    - -i eth0
  - -o : interface physique de sortie

Syntaxe

- Match spécifiques à TCP
  - --sport : port source
    - Unique : --sport 22 ou –sport !22
    - Multiple : --sport 22:80, --sport 1024:
  - --dport : port destination
  - --tcp-flags
    - Deux arguments : liste des flags à inspecter, liste des flags qui doivent être positionnés :
      - --tcp-flags SYN,ACK SYN : on inspecte les flags SYN et ACK, la règle correspond si le flag SYN est à 1
      - Mots-clefs : ALL et NONE.
        - --tcp-flags ALL NONE : tous les flags inspectés, doivent tous être à 0.
  - --syn : synmonyme de –tcp-flags SYN,RST,ACK SYN

Syntaxe

- Match
  - Spécifiques à UDP
    - --dport
    - --sport
  - Spécifiques à ICMP
    - --icmp-type : numéro du type
  - Autres types
    - Types dits Explicites. Doivent être introduits par -m type
    - Filtrage sur adresses MAC : -m mac
      - --mac-source
    - Filtrage sur état : -m state
      - --state RELATED/ESTABLISHED, etc.

Syntaxe

- Principales cibles
  - ACCEPT, DROP, REJECT : arrêt du traitement.
    - Si un paquet correspond à une règle qui utilise ces cibles, l'action de la cible est appliquée et les autres règles ne sont pas utilisées.
    - Conséquence : c'est la première règle qui est applicable qui est utilisée (first match first win).
    - Exception : LOG
  - DNAT, SNAT, MASQUERADE : modification des adresses source ou destination
  - REDIRECT : modification de l'adresse et/ou port destination

Syntaxe

- Exemple de règles IPTables :
  - iptables -A INPUT -p tcp -s 192.168.1.1 -d any -j ACCEPT
  - -A INPUT : on ajoute la règle à la chaine INPUT
  - -p tcp : Protocole TCP
  - -s : adresse IP source
  - -d : adresse Destination
  - -j ACCEPT : cible de la chaine.

Remarques

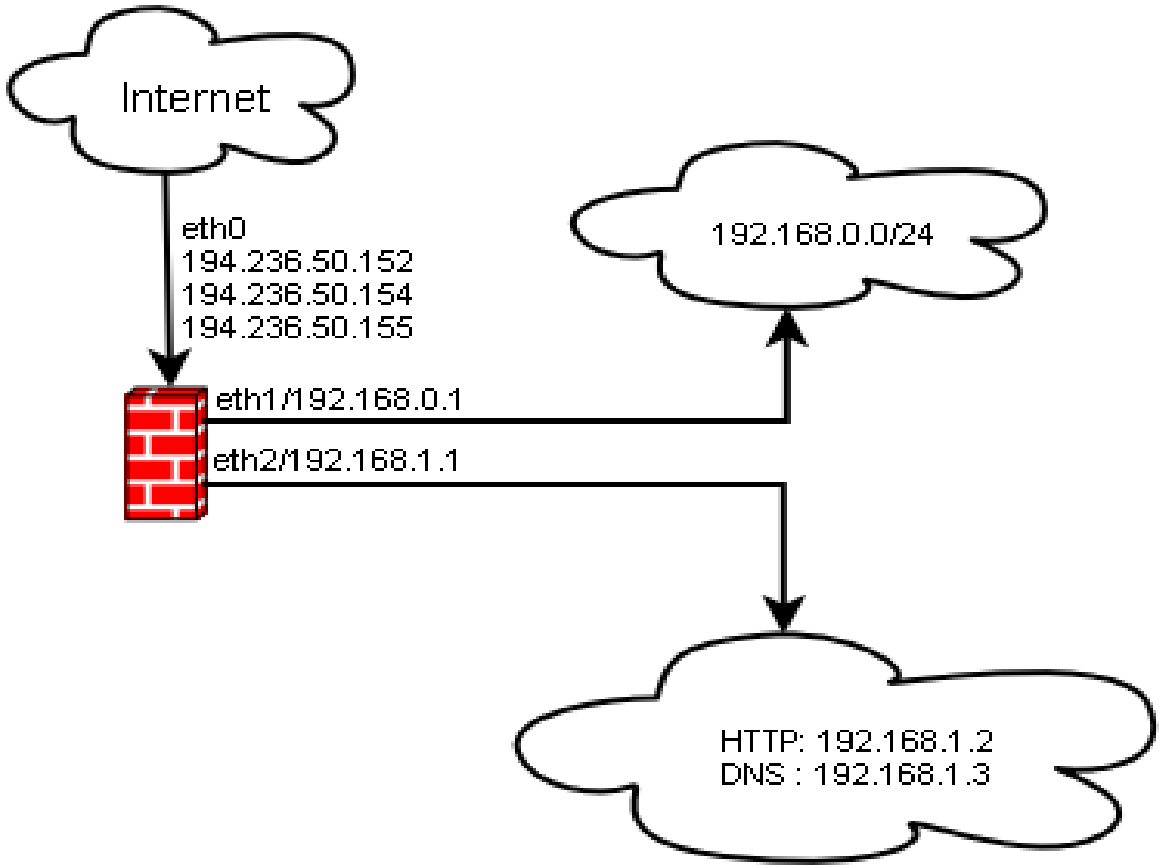
- Pour construire son fichier de règles :
  - Choisir une politique générale pour chaque chaine
  - Identifier tous les paramètres qui peuvent être passés sous forme de variables.
  - Choisir les chaines définies qui seront rattachées aux chaines prédéfinies.

Exemple de politique de sécurité

- Flux autorisés
  - Tous les flux depuis le réseau interne vers la DMZ
  - Tous les flux TCP depuis le réseau interne vers Internet.
  - Les flux SMTP vers le serveur de messagerie en DMZ
  - Les flux DNS vers le resolver en DMZ
  - Les flux HTTP et HTTPS vers Internet depuis le proxy en DMZ
- Flux interdits
  - Tous les flux entrants qui ne sont pas des réponses à des connexions ouvertes depuis la DMZ ou le réseau Interne
  - Les connexions directes HTTP et HTTPS depuis le réseau interne vers Internet
  - L'envoi de mail par SMTP depuis le réseau interne vers Internet
  - Les flux entrants d'Internet vers le réseau interne
  - Le trafic usurpé

Cas pratique

- Correspond à un réseau à trois zones : Internet, DMZ, LAN.



Etude de cas

- Script shell
  - Utilise la commande iptables pour charger les règles
  - Permet d'utiliser des variables.
  - #!/bin/sh
  - ##
  - # 1. Configuration options.
  - # 1.1 Internet Configuration.
  - INET_IP="194.236.50.152"
  - HTTP_IP="194.236.50.153"
  - DNS_IP="194.236.50.154"
  - INET_IFACE="eth0"

Etude de cas

- # 1.2 Local Area Network configuration.
- LAN_IP="192.168.0.1"
- LAN_IFACE="eth1"
- LAN_IP="192.168.0.11"
- # 1.3 DMZ Configuration.
- DMZ_HTTP_IP="192.168.1.2"
- DMZ_DNS_IP="192.168.1.3"
- DMZ_IP="192.168.1.1"
- DMZ_IFACE="eth2"
- # 1.4 Localhost Configuration.
- LO_IFACE="lo"
- LO_IP="127.0.0.1"

Etude de cas

- # 1.5 IPTables Configuration.
- IPTABLES="/usr/sbin/iptables"
- # 2. Module loading.
- # Needed to initially load modules
- /sbin/depmod -a
- # 2.1 Required modules
- /sbin/modprobe ip_tables
- /sbin/modprobe ip_conntrack
- /sbin/modprobe iptable_filter
- /sbin/modprobe iptable_mangle
- /sbin/modprobe iptable_nat
- /sbin/modprobe ipt_LOG
- /sbin/modprobe ipt_limit
- /sbin/modprobe ipt_state

Etude de cas

- # 3.1 Required proc configuration
- echo "1" > /proc/sys/net/ipv4/ip_forward
- # 4. rules set up.
- # 4.1 Filter table
- # 4.1.1 Set policies
- $IPTABLES -P INPUT DROP
- $IPTABLES -P OUTPUT DROP
- $IPTABLES -P FORWARD DROP
- # 4.1.2 Create userspecified chains
- # Create chain for bad tcp packets
- $IPTABLES -N bad_tcp_packets
- # Create separate chains for ICMP, TCP and UDP to traverse
- $IPTABLES -N allowed
- $IPTABLES -N icmp_packets

Etude de cas

- # 4.1.3 Create content in userspecified chains
- # bad_tcp_packets chain
- $IPTABLES -A bad_tcp_packets -p tcp --tcp-flags SYN,ACK SYN,ACK -m state --state NEW -j REJECT --reject-with tcp-reset
- $IPTABLES -A bad_tcp_packets -p tcp ! --syn -m state --state NEW -j LOG --log-prefix "New not syn:"
- $IPTABLES -A bad_tcp_packets -p tcp ! --syn -m state --state NEW -j DROP
- # allowed chain
- $IPTABLES -A allowed -p TCP --syn -j ACCEPT
- $IPTABLES -A allowed -p TCP -m state --state ESTABLISHED,RELATED -j ACCEPT
- $IPTABLES -A allowed -p TCP -j DROP

Etude de cas

- # ICMP rules
- # Accept Echo request & TTL = 0 warnings
- $IPTABLES -A icmp_packets -p ICMP -s 0/0 --icmp-type 8 -j ACCEPT
- $IPTABLES -A icmp_packets -p ICMP -s 0/0 --icmp-type 11 -j ACCEPT
- # 4.1.4 INPUT chain
- # Bad TCP packets we don't want
- $IPTABLES -A INPUT -p tcp -j bad_tcp_packets
- # Packets from LAN, DMZ or LOCALHOST
- # From DMZ Interface to DMZ firewall IP
- $IPTABLES -A INPUT -p ICMP -i $DMZ_IFACE -d $DMZ_IP -j ACCEPT

Etude de cas

- # From LAN Interface to LAN firewall IP
- $IPTABLES -A INPUT -p ICMP -i $LAN_IFACE -d $LAN_IP -j ACCEPT
- # From Localhost interface to Localhost IP's
- $IPTABLES -A INPUT -p ALL -i $LO_IFACE -s $LO_IP -j ACCEPT
- $IPTABLES -A INPUT -p ALL -i $LO_IFACE -s $LAN_IP -j ACCEPT
- $IPTABLES -A INPUT -p ALL -i $LO_IFACE -s $INET_IP -j ACCEPT
- # SSH Access to firewall from LAN
- $IPTABLES -A INPUT -p TCP -i $LAN_IFACE -s $ADMIN_IP –sport 1024: -d $LAN_IP –dport 22 -m state –state NEW -j ACCEPT

Etude de cas

- # 4.1.5 FORWARD chain
- # Bad TCP packets we don't want
- $IPTABLES -A FORWARD -p tcp -j bad_tcp_packets
- # DMZ section
- # General rules
- $IPTABLES -A FORWARD -i $DMZ_IFACE -o $INET_IFACE -j ACCEPT
- $IPTABLES -A FORWARD -i $INET_IFACE -o $DMZ_IFACE -m state --state ESTABLISHED,RELATED -j ACCEPT
- $IPTABLES -A FORWARD -i $LAN_IFACE -o $DMZ_IFACE -j ACCEPT
- $IPTABLES -A FORWARD -i $DMZ_IFACE -o $LAN_IFACE -m state --state ESTABLISHED,RELATED -j ACCEPT

Etude de cas

- # HTTP server
- $IPTABLES -A FORWARD -p TCP -i $INET_IFACE -o $DMZ_IFACE -d $DMZ_HTTP_IP --dport 80 -j allowed
- $IPTABLES -A FORWARD -p ICMP -i $INET_IFACE -o $DMZ_IFACE -d $DMZ_HTTP_IP -j icmp_packets
- # DNS server
- $IPTABLES -A FORWARD -p TCP -i $INET_IFACE -o $DMZ_IFACE -d $DMZ_DNS_IP --dport 53 -j allowed
- $IPTABLES -A FORWARD -p UDP -i $INET_IFACE -o $DMZ_IFACE -d $DMZ_DNS_IP --dport 53 -j ACCEPT
- $IPTABLES -A FORWARD -p ICMP -i $INET_IFACE -o $DMZ_IFACE -d $DMZ_DNS_IP -j icmp_packets

Etude de cas

- # LAN section
- $IPTABLES -A FORWARD -i $LAN_IFACE -j ACCEPT
- $IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
- # Log weird packets that don't match the above.
- $IPTABLES -A FORWARD -m limit --limit 3/minute --limit-burst 3 -j LOG --log-level DEBUG --log-prefix "IPT FWARD packet died: "
- # 4.1.6 OUTPUT chain
- # Bad TCP packets we don't want.
- $IPTABLES -A OUTPUT -p tcp -j bad_tcp_packets
- # SSH Access from Admin Station
- $IPTABLES -A OUTPUT -p TCP -o $LAN_IFACE -s $LAN_IP –sport 22 -d $ADMIN_IP –dport 1024: -m state –state ESTABLISHED, RELATED -j ACCEPT

Etude de cas

- # 4.2 nat table
- # 4.2.4 PREROUTING chain
- $IPTABLES -t nat -A PREROUTING -p TCP -i $INET_IFACE -d $HTTP_IP --dport 80 -j DNAT --to-destination $DMZ_HTTP_IP
- $IPTABLES -t nat -A PREROUTING -p TCP -i $INET_IFACE -d $DNS_IP --dport 53 -j DNAT --to-destination $DMZ_DNS_IP
- $IPTABLES -t nat -A PREROUTING -p UDP -i $INET_IFACE -d $DNS_IP --dport 53 -j DNAT --to-destination $DMZ_DNS_IP
- # 4.2.5 POSTROUTING chain
- # Enable simple IP Forwarding and Network Address Translation
- $IPTABLES -t nat -A POSTROUTING -o $INET_IFACE -j SNAT --to-source $INET_IP

Etude de cas

- Remarques sur l'activation du routage
- Deux méthodes :
  - Activation par sysctl.conf
    - Le routage est actif indépendamment du filtrage.
  - Activation dans le script de chargement des règles
    - Bonne solution car si les règles ne sont pas correctement chargées, le routage n'est pas activé.
    - Meilleure solution encore : activer le routage en fin de chargement des règles ou après avoir positionné les politiques par défaut à DROP sur les chaines INPUT/OUTPUT/FORWARD

Références

- NetFilter/IPTables
  - Tutoriel (anglais) : http://iptables-tutorial.frozentux.net
  - Site officiel - http://www.netfilter.org