Filtrage

Gaétan Richard & Alexandre Niveau

M2 e-Secure

6 octobre 2014

Adapté des cours de Jean Saquet
et du livre « Réseaux » d'Andrew Tanenbaum

Plan

Introduction

- Internet était au départ destiné à relier des universités
- Pas de sécurité :
  - données voyageant en clair, y compris les mots de passe
  - tout paquet peut transiter d'une adresse IP à une autre
- Solution contre les abus : rappel à l'ordre par la communauté
- Généralisation de l'Internet au grand public, applications commerciales : inefficacité du système basé sur la confiance

→ Nécessité de systèmes automatiques pour se protéger des attaques, et des erreurs involontaires

Classification des attaques

- On classe les attaques possibles en termes de services que l'on veut protéger :
  - confidentialité des échanges
  - intégrité des données
  - authentification des correspondants
  - non-répudiation des transactions
  - disponibilité des serveurs

Attaques : moyens et parades

- Accès physique aux machines
  → sécuriser l'accès aux locaux
- Écoute du réseau (ex : Wi-Fi)
  → chiffrage des données
- Failles des systèmes ou logiciels
  → mises à jour fréquentes
- Inondation (déni de service)
  → redondance
- Scan de ports pour repérer un logiciel attaquable
  → pare-feu

Attaques : moyens et parades

- Accès physique aux machines
  → sécuriser l'accès aux locaux
- Écoute du réseau (ex : Wi-Fi)
  → chiffrage des données + pare-feu
- Failles des systèmes ou logiciels
  → mises à jour fréquentes + pare-feu
- Inondation (déni de service)
  → redondance + pare-feu
- Scan de ports pour repérer un logiciel attaquable
  → pare-feu

Plan

Pare-feu

- Pare-feu (*firewall*) : logiciel permettant de filtrer les paquets entrants ou sortants d'une machine
- Exemples :
  - rejet de tout paquet venant de telle adresse IP
  - seul le trafic sur le port 80 est autorisé
  - traduction des adresses privées
- Principe de base : suite ordonnée de règles, de la plus particulière à la plus générale
- Dernière règle : refuser tout ce qui n'est pas explicitement autorisé

Types de pare-feu

- Différentes catégories de pare-feu, selon leur génération et leur objectif :
  - pare-feu sans état (*stateless*) : regarde chaque paquet indépendamment et le compare aux règles
  - pare-feu à états (*stateful*) : vérifie la conformité des paquets à une connexion en cours
  - pare-feu applicatif : vérifie la conformité des paquets au protocole attendu

Filtrage

- Le filtrage nécessite l'analyse des éléments de protocole. Il peut agir au niveau réseau, transport et/ou application
- Pour une question d'efficacité, on analyse les trois couches en même temps.
- Critères de filtrage : portent sur les différents paramètres des en-têtes des couches
  - adresses IP (3)
  - fonction ICMP (3)
  - ports sources et destination (4)
  - connexions entrantes ou sortantes (4)
  - protocole de transport (4)
  - données (7)
- Le filtrage porte alors séparément sur chaque paquet arrivant à, partant de ou traversant la machine surveillée.

Filtrage niveau réseau

- Les messages transitent dans des datagrammes IP, et doivent traverser les routeurs
- Le routeur peut accepter ou non de router le datagramme en fonction des adresses IP (source et destinataire)
- Cependant, il est assez facile d'usurper une adresse IP source ; ce filtrage empêche l'établissement d'une connexion, mais pas l'envoi de datagrammes, qui peuvent suffire pour certaines attaques

Filtrage niveau transport

- Les messages sont le plus souvent des segments TCP ou des datagrammes UDP (ou messages ICMP).
- Analyse de niveau transport souvent faite au niveau des routeurs : les messages doivent être analysés au niveau réseau, autant en profiter pour analyser aussi le niveau transport

→ contrôle de plusieurs paramètres, de niveau IP et de niveau transport

- Non-respect des principes de « bout-en-bout » et d'indépendance des couches

Filtrage niveau application

- Vérification de la conformité des messages au protocole : par exemple, que c'est bien du HTTP qui passe sur le port 80 et pas autre chose (tunnel)
- Vérification des données elles-mêmes : par exemple, analyse du contenu d'un courriel pour détecter le spam ou les virus connus

Filtrage avec suivi, ou surveillant les répétitions

- Il peut être intéressant de surveiller les successions de paquets plutôt que les paquets isolés pour par exemple :
  - Repérer les tentatives d'inondation
  - Ne laisser passer que des flots de données relatifs à une connexion dont l'ouverture a été autorisée
  - Laisser passer les réponses aux requêtes qui ont été autorisées

$\longrightarrow$ nécessité de se souvenir de ce qui s'est passé : filtrage avec état

Où filtrer ?

- Classiquement : passerelle entre Internet et réseau interne, car toutes les communications avec l'extérieur doivent y passer
- Attention à la charge du routeur

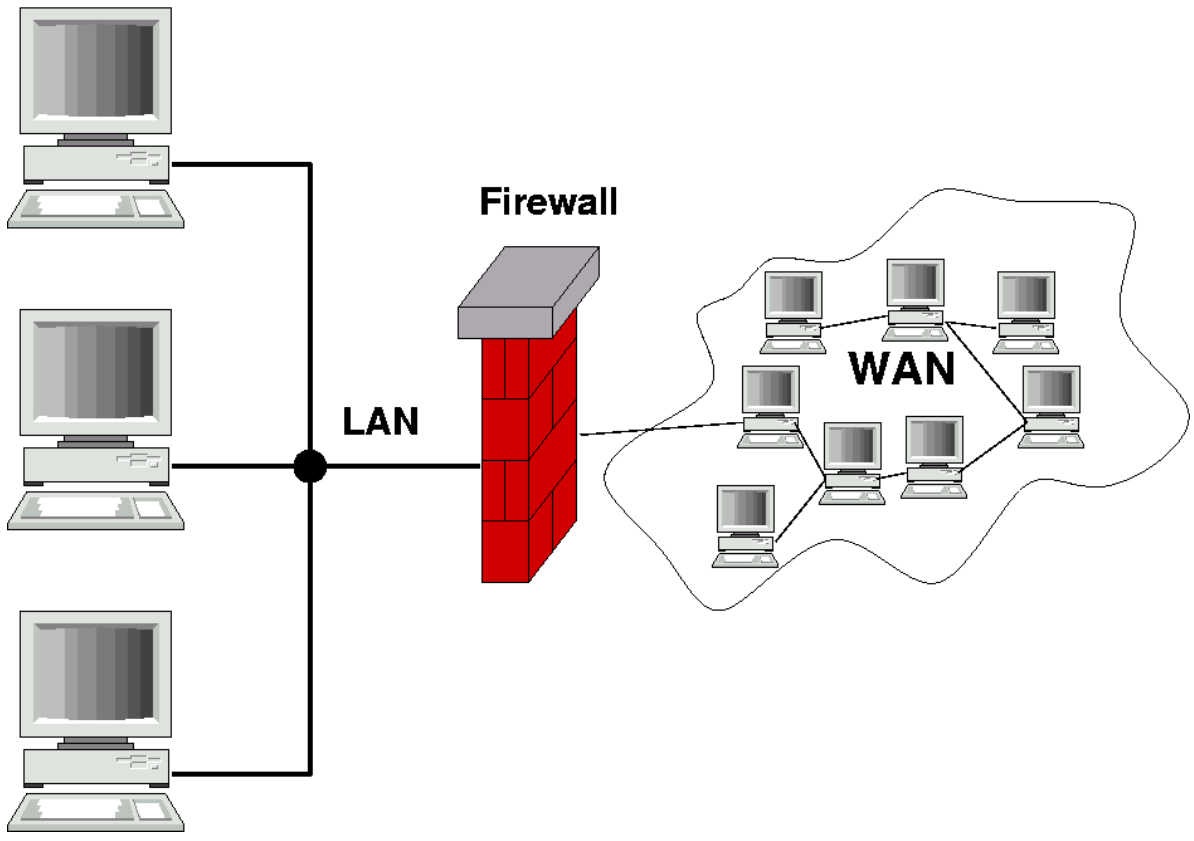
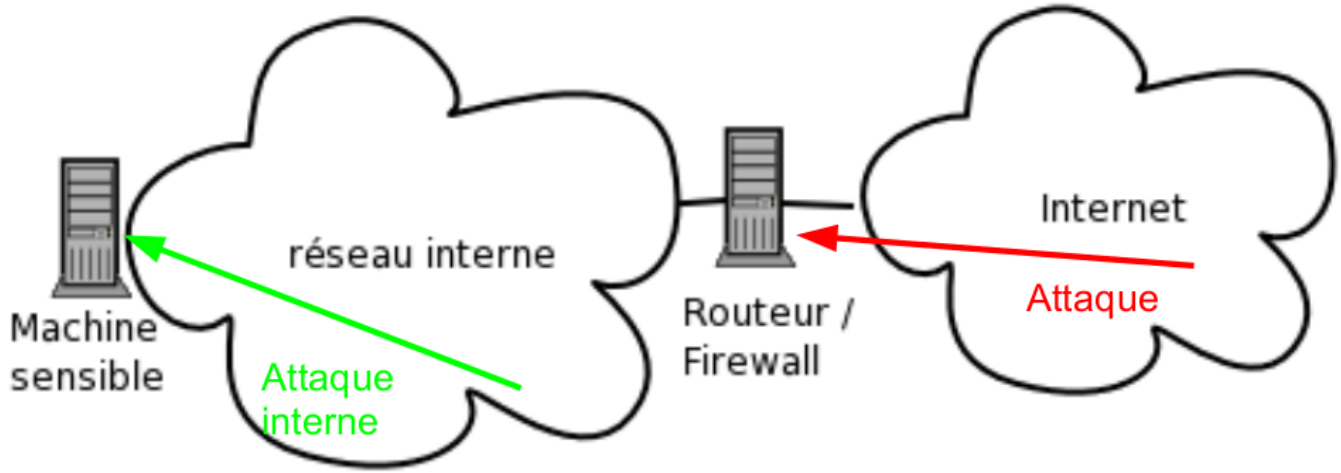


Image : https://commons.wikimedia.org/wiki/File:Gateway_firewall.png

Protéger de l'extérieur ne suffit pas

- La centralisation du pare-feu est pratique au niveau de la gestion, mais ne protège pas contre les risques internes.



⟶ « auto-firewalls » sur les machines sensibles (serveurs de données confidentielles par exemple) adaptés à chaque cas

⟶ séparation du réseau des utilisateurs et de celui des machines sensibles

Configuration

- Classiquement :
  - auto-firewall « standard » pour les postes de travail ordinaires
  - auto-firewall particulier pour chaque serveur, chaque machine ayant une fonction particulière
  - un ou plusieurs pare-feu généraux entre réseaux internes et externe, et pour l'accès à la zone des serveurs publics → « zone démilitarisée » (DMZ)

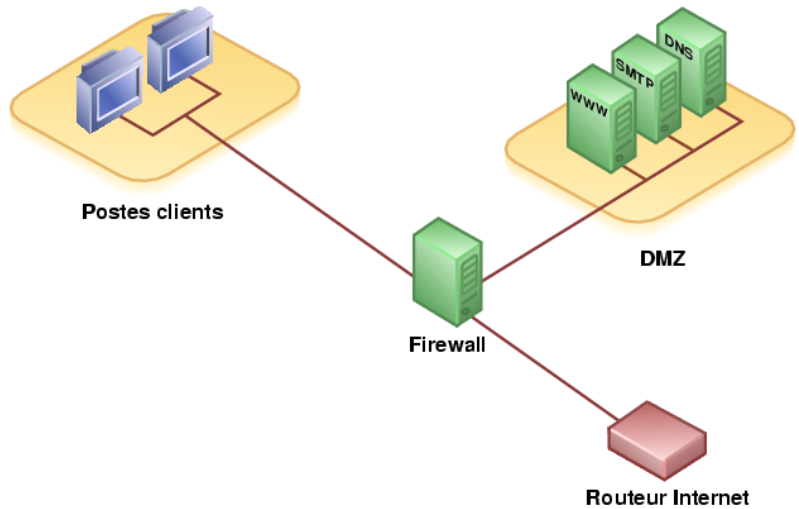


Image : `https://commons.wikimedia.org/wiki/File:Demilitarized_Zone_Diagram.png`

Plan

Traduction d'adresses

- Un routeur fait de la traduction d'adresses (NAT, *Network Address Translation*) lorsqu'il modifie les adresses dans les datagrammes IP pendant le routage
- Objectif : remapper un espace d'adressage dans un autre
  - interconnexion de réseaux avec des adressages incohérents
  - accès à Internet pour un réseau avec adressage privé

NAT et adresses privées

- Plages d'adresses privées IPv4 (RFC 1918) :
  - 10.0.0.0/8
  - 172.16.0.0/12
  - 192.168.0.0/16
- Utilisées dans les LAN, mais non routées sur Internet
- Objectifs : pouvoir utiliser la pile TCP/IP dans son réseau interne sans consommer des adresses publiques
- Manque d'adresses IPv4 $\rightarrow$ solution temporaire : utilisation des adresses privées et traduction d'adresses pour permettre aux hôtes du LAN d'accéder à Internet

NAT basique

- Correspondance directe entre les adresses de deux espaces d'adressage
- Une machine M a une IP A, mais un réseau voisin la voit avec l'IP B
- Le routeur
  - change A en B dans l'adresse source des datagrammes de M qu'il route vers le réseau voisin
  - change B en A dans l'adresse destination des datagrammes vers M qu'il route depuis le réseau voisin
- Dans chaque cas il doit recalculer le *checksum* IP et les *checksums* des couches supérieures utilisant l'adresse IP (TCP, UDP...)

Mascarade

- Utilisation la plus courante : cacher un réseau IP privé derrière une seule adresse IP publique
- Ne peut pas fonctionner comme le NAT basique : ambiguïté pour les réponses

→ Le routeur doit modifier d'autres informations dans les paquets sortants pour pouvoir identifier le destinataire des réponses à ces paquets

- Typiquement, il change le numéro de port TCP/UDP et maintient une table de traduction.
- Cette technique s'appelle NAPT (*network address and port translation*, RFC 2663), ou plus couramment « mascarade IP » (*IP masquerading*)

Redirection de port

- Avec NAPT, une connexion ne peut être initiée que par une machine interne

→ impossible d'utiliser une machine interne comme serveur pour l'extérieur

- Redirection de port (*port forwarding*) : on pré-remplit la table de traduction du routeur pour que les paquets de l'extérieur sur tel port du routeur soient redirigés vers telle machine interne

Terminologie ambiguë

- Selon la RFC 2663
  - *static address assignment* : autant d'adresses publiques que privées
  - *dynamic address assignment* : plusieurs adresses publiques parmi lesquelles le routeur peut choisir
- Mais d'autres sources utilisent statique/dynamique pour la gestion des ports
  - *static NAT* : redirection de port (la table de traduction est statique)
  - en français *NAT dynamique* fait référence à NAPT, les ports sont choisis dynamiquement (en particulier la mascarade est un cas particulier de NAT dynamique)
- SNAT et DNAT peuvent signifier statique/dynamique ou source/destination
- Conclusion : attention au contexte...

Avantages de NAT

- Économie d'adresses IPv4
- Pas de limitation due aux adresses pour la configuration du réseau privé
- Plan d'adressage privé invisible depuis l'extérieur

→ mais ne pas compter dessus pour la sécurité

Problèmes de NAT

- NAT rejette une connexion entrante non sollicitée : problématique par exemple pour les protocoles pair-à-pair
- Adresse IP ou numéro de port transmis dans le paquet (FTP) : l'information n'est plus valide après traduction
  $\rightarrow$ le routeur doit inspecter le contenu
- Fragmentation
- Protocoles exigeant un port fixe
- IPsec : en-tête authentifié invalide
- De manière générale : ne respecte pas les principes du modèle en couches

Plan

Plan

Introduction

- 1996, noyau Linux 2.0 : ipfwtk (*IP firewall toolkit*)
- 1998, noyau Linux 2.2 : ipchains, plus de fonctions, mais toujours sans état (*stateless*)
- 2001, noyau Linux 2.4 : Netfilter
- Très efficace et rapide, plus complet que ipchains (permet notamment le suivi de connexion).

iptables

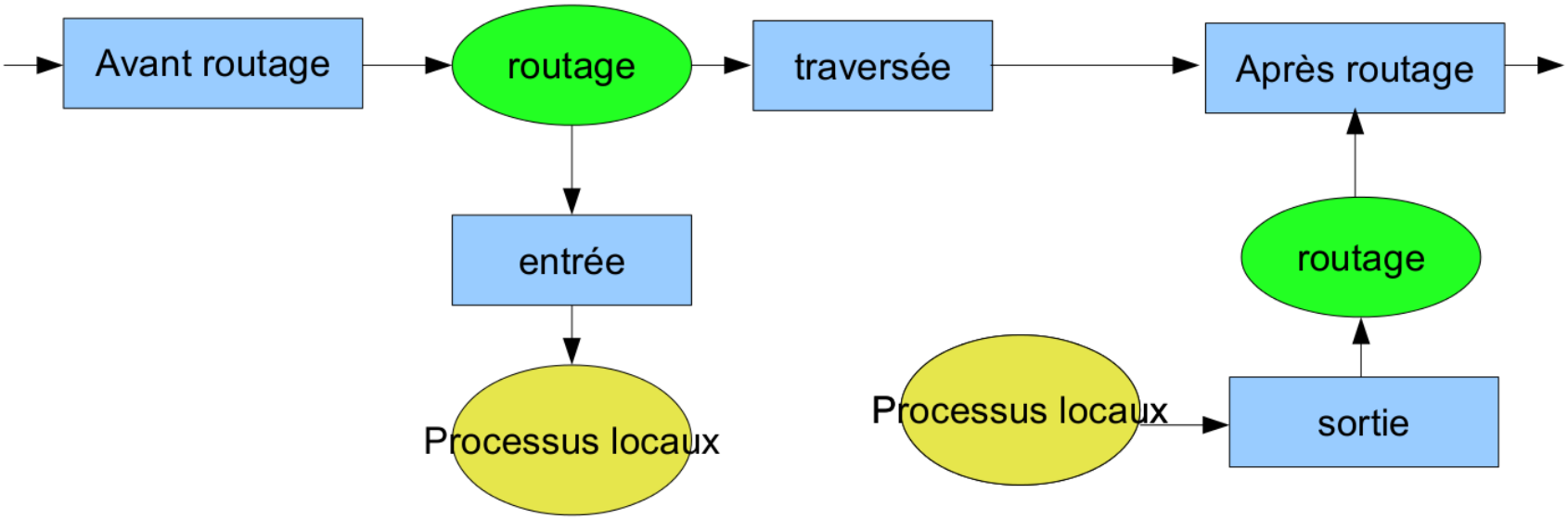
- Outil de configuration de Netfilter
- Classiquement, Netfilter peut filtrer le paquets comme les transformer (traduction d'adresse ou/et de port).
- Les deux fonctions sont différentes, les règles qui les gouvernent sont donc séparées dans différentes tables, en particulier `filter` et `nat`.
- Chacune comporte des « chaînes » (listes de règles) se positionnant à un moment bien précis de l'analyse des paquets : en entrée, sortie, ou traversée (pour `filter`), avant ou après routage (pour `nat`)...

Principe de Netfilter

- Les datagrammes IP peuvent être analysés à différents stades de leur transit dans une machine :
  - à l'arrivée, avant toute opération de routage
  - après décision de routage, et avant
    - remise à la couche supérieure (si datagramme arrivé à destination)
    - envoi à passerelle suivante (si datagramme doit être routé)
  - après création du datagramme (si généré par la machine depuis couche supérieure)
  - après toutes les opérations de routage, si le datagramme sort de la machine

Schéma simplifié de Netfilter

En bleu : points de filtrage possibles d'un datagramme et/ou d'action sur ce dernier



- la table « filter » utilise les chaînes INPUT, OUTPUT et FORWARD
- la table « nat » utilise PREROUTING, POSTROUTING et OUTPUT.

Iptables, utilisation

- Si on se limite à la table `filter`, on devra donc définir des règles dans les chaînes :
  - INPUT pour les paquets destinés à la machine
  - OUTPUT pour les paquets issus de la machine
  - FORWARD pour les paquets traversant la machine
- Pour la table `nat`, on utilisera les chaînes PREROUTING, POSTROUTING, éventuellement OUTPUT

Ordre des règles

- À chaque étape, netfilter essaye d'appliquer la première règle de la chaîne ; si applicable, il s'arrête
- Si pas applicable, il passe à la suivante dans la chaîne, etc.
- Si plus de règle, il utilise la politique par défaut de la chaîne
- L'ordre des règles est donc fondamental. Exemple :
  1. autoriser la machine d'IP x à se connecter au port 22 de la machine d'IP y
  2. interdire à toute machine de se connecter au port 22 de la machine d'IP y
- Si on inverse l'ordre, la machine x ne pourra pas se connecter, car la règle générale (devenue la première) se sera appliquée avant la règle particulière

Autres tables

- `mangle` : permet de modifier certains champs des paquets (TTL, type of service). Peut être utilisée en conjonction avec des règles de routage spécifiques.
- `raw` : marquage spécifique des paquets pour éviter leur prise en compte par le suivi de connexion *conntrack* (voir plus loin)

Schéma complet

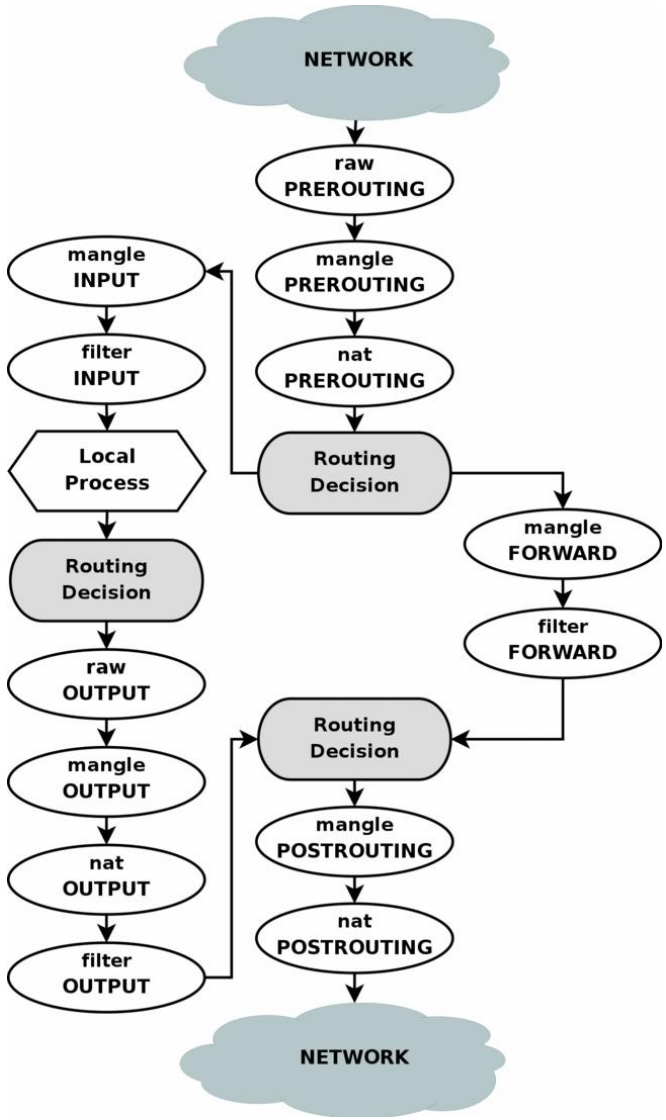


Image : `http://www.iptables.info/en/structure-of-iptables.html`

Plan

Syntaxe de base

- La syntaxe générale est :
  `iptables [-t table] command [match] [target/jump]`
- table : par défaut, `filter`
- command ajoute des règles, les supprime, les liste, etc.
- match : suite de critères de filtrage (adresse, port source/dest, protocole, interface...)
- target/jump : action à effectuer si la règle s'applique

Commande

- iptables prend en paramètre une commande (opération) :
  - -N ou -X ou -L ou -F pour créer une chaîne, la détruire, lister ses règles, éliminer ses règles
  - -A, -D ou -R pour ajouter, supprimer ou modifier une règle dans une chaîne
  - -P pour définir la politique par défaut d'une chaîne

Ajout de règle

- Pour ajouter une règle à une chaîne :

```
iptables [-t table] -A chaîne [-p proto]
[-s source] [-d destination]
[--sport portsource] [--dport portdest]
[-i ifsource] [-o ifdest] [-j cible]
```

- cible définit l'opération à appliquer au paquet :
  - ACCEPT, DROP, REJECT pour les chaînes de « filter »
  - SNAT, DNAT, MASQUERADE pour les chaînes de « nat »

Exemples d'ajout de règles

- `iptables -A INPUT -p TCP -s 192.168.1.0/24 --dport 80 -j ACCEPT`
  accepte les connexions sur le port 80 (http) de la machine, venant d'une machine d'adresse commençant par 192.168.1
- `iptables -t nat -A postrouting -s 192.168.0.0/24 -o eth1 -j MASQUERADE`
  Opère la translation d'adresse pour tout paquet émis par une machine 192.168.0.xx et sortant par l'interface eth1

Plan

Extensions d'iptables

- De nombreux paramètres de filtrage sont définis dans des extensions
- Nécessitent d'ajouter `-m nomextension` (ou `--match nomextension`) pour pouvoir les utiliser
- Certaines sont ajoutées automatiquement, par exemple filtrer sur le protocole TCP charge automatiquement l'extension associée :
  `iptables -A INPUT -p tcp --sport 456 -j DROP`
  devrait s'écrire
  `iptables -A INPUT -p tcp -m tcp --sport 456 -j DROP`

Concordance d'état (conntrack)

- Suivi de connexion avec l'option `--state` de l'extension `state`
  - NEW paquet engendrant une nouvelle connexion
  - ESTABLISHED : le contraire
  - RELATED : relatif à une connexion (par ex., erreur ICMP, connexion de données FTP...)
  - INVALID : paquets non identifiés
- Exemple : suivi d'état de la connexion TCP, mais vaut aussi pour UDP ou ICMP
- Le suivi de connexion permet l'analyse plus fine des paquets afin, par exemple, de s'adapter à des protocoles ouvrant d'autres connexions (FTP, SIP...)
- Conntrack est toutefois un peu coûteux. On a parfois besoin de s'en passer, on peut alors utiliser des marquages avec la table raw (cible NOTRACK)

Autoriser les réponses aux requêtes

- Avec iptables, une seule règle pour que les réponses aux requêtes acceptées puissent passer en sens inverse : `iptables -A INPUT -m state --state ESTABLISHED, RELATED -j ACCEPT` (autorise les paquets entrants en réponse aux requêtes qu'on a laissées sortir ; fonctionne pour TCP et UDP)

Filtrage fin

- `--syn` (ext. `tcp`) : pour spécifier uniquement les demandes de connexion TCP
- `--icmp-type` (ext. `icmp`) : filtrage fin des paquets ICMP, par type, code... (`iptables -p icmp -h` pour la liste)
- `--mac-source` (ext. `mac`) : adresse mac
- `--limit` (ext. `limit`) : nb max de concordances par seconde (ou par minute, etc.)
  $\rightarrow$ permet de n'agir que sur quelques paquets (pour le log, par exemple) ou d'empêcher les paquets d'arriver trop vite (pour éviter les attaques répétées)
- etc.

Cibles particulières

- Chaînes utilisateur : `-j <nom chaine>`
  → renvoie l'analyse aux règles de la chaîne indiquée
- LOG log des paquets
- REJECT renvoie une erreur ICMP, par défaut « port unreachable »

NAT

- On peut traduire la source ou la destination avec les cibles SNAT et DNAT

→ nécessite de préciser la nouvelle adresse avec l'option `--to 192.168.168.192`

- MASQUERADE traduit l'adresse source en la remplaçant par celle du routeur : utile notamment si cette adresse est assignée dynamiquement

Autres outils de Netfilter

- ip6tables : outil de configuration de Netfilter pour les paquets IPv6

  $\rightarrow$ fonctionne de la même façon, à quelques détails près (protocole ICMP différent, pas besoin de NAT a priori...)
- arptables : pour filtrer les paquets ARP
- ebtables (*Ethernet bridge*) : pour faire un pare-feu sur un pont. Filtre les trames, au niveau liaison.

Plan

Présentation

- *Packet Filter*, ou pf, est l'équivalent OpenBSD de Netfilter pour Linux
- Pare-feu pouvant aussi faire du NAT, de la redirection de trafic, et de la gestion de bande passante
- Activé par défaut, mais avec une règle qui laisse tout passer

Contrôle

- Fichier de règles par défaut : `/etc/pf.conf`
- Commande `pfctl` pour le contrôler
  - `pfctl -e` ou `-d` : *enable, disable*
  - `pfctl -f <fichier>` : lecture d'un fichier de règles
  - `pfctl -sr`, `-ss`, `-si`, `-sa` : affichage des règles, des tables d'état, des statistiques, de tout

Principe

- On bloque ou on laisse passer : *block* ou *pass*
- Nuance *drop/reject* de netfilter : *block drop* ou *block return*
- Règles ordonnées, mais par défaut toutes les règles sont évaluées et la dernière qui correspond s'applique
- Option *quick* pour stopper l'évaluation sur une règle particulière

Paramètres

```
action [<direction>] [log] [quick] [on <iface>] [af]
[proto <protocol>] [from <src_addr> [port <src_port]]
 [to <dst_addr> [port <dst_port>]] [flags <tcp_flas>]
 [state]
```

- direction : `in` ou `out`
- `af` : `inet` ou `inet6`
- `proto` : `tcp`, `udp`, `icmp`, `icmp6`, numéro, mnémonique de `/etc/protocols`, ou liste

État

- pf peut conserver l'état des connexions dans une table
- On teste si un paquet correspond à une connexion déjà autorisée → pas de règle pour le retour
- Par défaut, toute règle *pass* crée une entrée dans la table d'états
- `state` a plusieurs options : `keep` (défaut), `no`, `modulate` (gestion des numéros de séquence initiaux TCP), `synproxy` (proxy TCP)

État : précisions

- Pour UDP, on peut configurer un *timeout*
- Il existe des options de suivi pour le nombre max d'entrées dans la table, pour les adresses IP, etc.
- Exemple :
  `pass in on eth0 proto tcp to 134.15.63.13 port www keep state (max 200, source-track rule, max-src-nodes 100, max-src-states 3)`
  (total 200 états, 100 clients, 3 connexions par client)

Filtrage fin

- Proxy TCP : permet d'éviter l'inondation de paquets SYN (à utiliser avec modération)
- Blocage de paquets avec adresse usurpée
  - adresse arrivant via une mauvaise interface
  - route de retour vers cette adresse ne correspondant pas à l'arrivée
- Reconnaissance d'empreintes de connexions TCP : permet de reconnaître le système d'exploitation grâce à certaines caractéristiques dans les paquets SYN

```
pass  in on $ext_if proto tcp from any os OpenBSD keep
    state
block in on $ext_if proto tcp from any os "Windows 2000"
block in on $ext_if proto tcp from any os "Linux 2.4 ts"
block in on $ext_if proto tcp from any os unknown
```

- Filtrage des options IP

Traduction d'adresses

- Option `nat-to` dans une règle `pass` :
  `pass out on tl0 from 192.168.1.0/24 to any nat-to 198.51.100.1` ou mieux
  `pass out on tl0 inet from dc0:network to any nat-to (tl0)`
- Ou règle `match` + règle `pass`
  - `match` définit le NAT à appliquer pour des paquets vérifiant certaines conditions
  - si une règle `pass` est rencontrée avec les mêmes conditions, on n'applique pas le NAT au paquet

→ permet de faire des exceptions à la règle NAT

- NAT permanent : `binat-to`, ports non modifiés
- Redirection : `rdr-to`
  `pass in on tl0 proto tcp from any to any port 80 rdr-to 192.168.1.20`

Divers

- Pour simplifier l'écriture des règles, on peut utiliser des variables, des macros, des listes d'adresses, de ports, etc.
- Syntaxe assez souple, et peut souvent être simplifiée
- Pour plus de détails : `http://www.openbsd.org/faq/pf/fr`
- pfsense est une distribution OpenBSD configurée comme routeur et pare-feu, basée sur pf