|  |  |  |
| --- | --- | --- |
| iptables  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png  Iptables (suite)    http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png  iptables (suite)  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png  iptables (suite)  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png  iptables (suite)  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png  iptables (suite)  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png  iptables (suite)  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png  iptables (suite)  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png  iptables (suite)  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png  iptables (suite)  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png  iptables (suite)  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png  iptables (suite)  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png  iptables (suite)  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png  iptables (suite)  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png |  | Pare-feu sous Linux  **iptables**  Introduction  Le **pare-feu le plus populaire** utilisé sous Linux est iptables.  Il propose une **interface en ligne de commande** permettant de configurer netfilter. Il permet d'établir un certain nombre de **règles pour dire par quels ports on peut se connecter à un hôte** (ordinateur) mais aussi à quels **ports on a le droit de se connecter**.  **Netfilter** est un module du noyau Linux qui **offre la possibilité de contrôler**, **modifier et filtrer les paquets IP**, et de **suivre les connexions**. Il fournit ainsi les **fonctions de pare-feu**, **de partage de connexions internet** et d'**autorisation** d**u trafic réseau**.  *iptables* est un logiciel libre de l'espace utilisateur Linux grâce auquel l'administrateur système peut configurer les chaînes et règles dans le pare-feu en espace noyau (et qui est composé par des modules Netfilter).  Différents programmes sont utilisés selon le protocole employé : **iptables est utilisé pour le protocole IPv4**, **Ip6tables pour IPv6,** **Arptables pour ARP** (*Address Resolution Protocol*) ou encore **Ebtables**, spécifique aux **trames Ethernet**.  Un pare-feu (ou firewall) permet de bloquer l'accès à certains ports  Par exemple, si on veut empêcher toute connexion FTP (considérant que FTP n'est pas sûr), il est possible de bloquer le port 21 (port TCP utilisé par FTP).  En général, la technique ne consiste pas à bloquer certains ports mais plutôt à bloquer par défaut **tous** les ports et à en autoriser seulement quelques-uns.  Le but d'un pare-feu est d'empêcher que des programmes puissent communiquer sur le réseau sans autorisation. Aujourd'hui, même sous Windows (depuis Windows XP SP2), un pare-feu est intégré par défaut tant le problème est important.  Avoir un pare-feu ne prémunit pas contre les virus (bien que sous Linux, ils demeurent rares).  En revanche, cela rend la tâche **particulièrement difficile** aux pirates qui voudraient accéder à un hôte visible sur le réseau.  L’objectif de l’installation d’un pare-feu est de bloqu**er par défaut les ports et de n'autoriser seulement ceux dont nous avons besoin**, qui sont considérés comme « sécures » et qui sont utilisés.  Par exemple, le port TCP 80 utilisé pour le web est un port sûr que l’on peut activer.  **Notez**, et cela est important, **qu'il** **y a des portes d'entrée et des portes de sortie sur un** **hôte** (ce ne sont pas nécessairement les mêmes).  *iptables* existe aussi pour *ipv6*, pour cela il suffit d'utiliser la **commande ip6tables** au lieu de *iptables*.  Utilisation iptables est un programme extrêmement puissant… mais tout aussi complexe.  C’est pourquoi *iptables* n’est **paramétrable seulement en tant que *root***.  Pour la suite des opérations, il est donc **recommandé de basculer en superutilisateur** dès à présent :  **$ sudo su**  **La table *filter***  *iptables* **fonctionne selon un système de tables**, ces **tables sont composées de chaînes**.  Dans le cadre de la configuration et de l'utilisation de *netfilter* comme pare-feu, c'est la **table Filter qui est utile**, elle permet de **filtrer les paquets réseaux**.  Chaque paquet entrant est analysé afin de déterminer notamment sa source et sa destination.  Elle est composée de trois sortes de chaînes…  **INPUT** Permet d'**analyser** **les paquets entrants**.  Si le paquet est adressé au poste, il est confronté au filtre INPUT.  **FORWARD** Permet d'**analyser** et d'**autoriser** **les paquets à passer d'une interface à une autre**, seulement dans le cadre d'une interface réseau servant de passerelle.  **OUTPUT** Permet d'**analyser** **les paquets sortants**.  Si le paquet sort du poste, il passera par la chaîne OUTPUT.  À cette table, peuvent être affectées des politiques (*policy*) : **DROP**, **LOG**, **ACCEPT**, **SNAT**, **DNAT** et **REJECT**.  **iptables –L ou afficher les règles**  *iptables -L* **permet d’afficher les règles qui régissent actuellement le pare-feu**.  **# iptables -L**  Chain INPUT (policy ACCEPT)  target     prot opt source               destination  Chain FORWARD (policy ACCEPT)  target     prot opt source               destination  Chain OUTPUT (policy ACCEPT)  target     prot opt source               destination  **Par défaut, les règles sont vides**.  Il y a trois tableaux mais qui pour le moment ne contiennent aucune ligne.  Par ailleurs, à chaque fois les **mots policy ACCEPT** sont présents, cela signifie que **tout le trafic est accepté**.  Le **pare-feu est tout simplement inactif et il ne bloque rien**.  S’il y a **déjà des règles inscrites** pour le pare-feu (ce qui ne devrait pas être), il est **possible de les réinitialiser**. Ne pas faire que **s’il on est certain de vouloir le faire**. En effet, sur un hôte partagé, il est possible quelqu'il déjà configuré le pare-feu et il serait dommage de saboter tout son travail.  **# iptables –F** Attention… Cette **commande réinitialise toutes les règles *iptables***.  **Le principe des règles**  Voici ce que cela pourrait donner lorsqu'on aura établi des règles… par exemple pour la section INPUT :  **# iptables -L**  Chain INPUT (policy DROP)  target  prot opt  source destination  ACCEPT tcp   --   anywhere anywhere tcp dpt:www  ACCEPT tcp   --   anywhere anywhere tcp dpt:ssh  ACCEPT tcp   --   anywhere anywhere tcp dpt:pop3  Première chose à savoir : **l'ordre des règles est important**.  En effet, *iptables* les **lit de haut en bas** et la **position de ces règles influe sur le résultat final**.  Donc, les règles sont numérotées.  Pour afficher les numéros de lignes, ajoutez à la commande le commutateur --line-numbers  **# iptables -L --line-numbers**  Chain INPUT (policy DROP)  Num  target prot  opt  source       destination  1 ACCEPT      tcp   --   anywhere              anywhere            tcp dpt:www  **2 ACCEPT      tcp   --   anywhere           anywhere        tcp dpt:ssh**  3  ACCEPT      tcp  --   anywhere              anywhere            tcp dpt:pop3  Ainsi, la règle filtrant SSH est la règle numéro 2.  Chaque ligne correspond à une **règle différente qui permet de filtrer ou non une adresse IP ou un port**.  Parmi les colonnes intéressantes, on note :   * **target** – **ce que fait la règle**.  Ici c'est ACCEPT, c'est-à-dire que cette ligne autorise un port et / ou une adresse IP; * **prot** – **le protocole utilisé** (tcp, udp, icmp).  Il est bien de se rappeler que TCP est celui auquel on a le plus recourt. ICMP quant à lui permet à l’hôte de répondre aux requêtes de type « *ping* »; * **source** – **IP de source**.  Pour INPUT, la source est l'ordinateur distant qui se connecte à l’hôte; * **destination** – **IP de destination**.  Pour OUTPUT, c'est l'ordinateur auquel l’hôte se connecte; * ***la dernière colonne*** – indique le **port après les deux points** « : ».  Ce port est affiché en toutes lettres, mais avec -n vous pouvez obtenir le numéro correspondant.   Sur cet exemple, seuls les ports HTTP, SSH et POP3 (courriel) sont autorisés en entrée.  Personne ne peut se connecter à la machine par un autre biais. En effet, si on regarde bien, par défaut le pare-feu est configuré pour qu'il ignore tous les autres paquets: (policy DROP).  **L’ajout et la suppression des règles**  Voici les principales commandes à connaître.   * **-A ou --append <chaine> ajoute une règle en fin de liste** pour la chaine indiquée (INPUT ou OUTPUT, par exemple). * **-D ou --delete <chaine> <numéro>** **supprime la règle n° rulenum** pour la chaine indiquée. * **-I ou --insert <chaine> <numéro>** **insère une règle au milieu de la liste** à la position indiquée par *rulenum*.  Si **aucune position** rulenum n’est indiquée, la **règle sera insérée en premier**, tout en haut dans la liste. * **-R ou --replace <chaine> <numéro> remplace la règle n° rulenum** dans la chaine indiquée. * **-L ou --list** **Liste les règles**. * **-F ou --flush <chaine>** **vide toutes les règles de la chaine indiquée**. Cela revient à supprimer toutes les règles une par une pour cette chain. * **-P ou --policy <chaine> ou <cible>** **modifie la règle par défaut** pour la chaine… cela permet de dire, par exemple, que par défaut tous les ports sont fermés, sauf ceux que l'on a indiqués dans les règles.   De manière générale, l'ajout d'une règle se passe suivant ce schéma:  **iptables -A <chaine> -p <protocole> --dport <port> -j <décision>**  On remplace …   * **<chaine>** par la section qui vous intéresse (INPUT ou OUTPUT), * **<protocole>** par le nom du protocole à filtrer (TCP, UDP, ICMP…) * **<port>** qui désigne le port concerné et enfin * ***décision*** par la décision à prendre:  ACCEPT pour accepter le paquet,  REJECT pour le rejeter  ou bien DROP pour l'ignorer complètement.   Il existe également les paramètres LOG, SNAT et DNAT.  Le mieux est de découvrir comment on ajoute une règle par une série d'exemples.  **# iptables -A INPUT -p tcp --dport ssh -j ACCEPT**  Cela ajoute à la section INPUT (donc, pour le trafic entrant) une règle sur les données reçues à l’aide du protocole TCP sur le port de SSH (on peut remplacer ssh par son numéro du port, soit 22).  Lorsque l’hôte recevra des données en TCP sur le port de SSH, celles-ci seront acceptées ; cela permettra donc de se connecter à distance à l’hôte à l’aide du protocole SSH.  Il est possible de faire de même avec d'autres ports :  **#** **iptables -A INPUT -p tcp --dport www -j ACCEPT**  … pour le Web (80).  **#** **iptables -A INPUT -p tcp --dport imap4 -j ACCEPT**  … pour les courriels, ...  **Remarque** … Si on **ne précise pas le port (en omettant la section dport), tous les ports seront acceptés** !  Autoriser les pings  En plus d'autoriser le trafic sur ces ports, il est intéressant d'autoriser le protocole ICMP (pour pouvoir faire un ping) sur tous ces derniers:  **#** **iptables -A INPUT -p icmp -j ACCEPT**  Comme il n’est pas indiqué de section *--dport*, cette règle s'applique à tous les ports – **mais pour les pings (ICMP) uniquement** !  L’ordinateur répondra alors aux requêtes *ICMP echo* (*ping*) pour indiquer qu'il est bien en vie.  Les règles *iptables* pour INPUT devraient maintenant ressembler à ceci :  **#** **iptables -L**  Chain INPUT (policy ACCEPT)  target      prot  opt  source  destination  ACCEPT tcp   --   anywhere              anywhere             tcp dpt:www  ACCEPT  tcp   --   anywhere              anywhere             tcp dpt:ssh  ACCEPT tcp   --   anywhere              anywhere             tcp dpt:imap4  ACCEPT icmp  --   anywhere              anywhere  Autoriser les connexions locales et déjà ouvertes  Pour l'instant, les règles sont encore un peu trop restrictives et pas vraiment utilisables  (on risque de ne plus pouvoir faire grand-chose).  Il est possible d’ajouter deux règles pour assoupli un peu le pare-feu et le rendre enfin utilisable.  **#** **iptables -A INPUT -i lo -j ACCEPT**  **#** **iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT**  Ces deux règles utilisent des options un peu différentes.  Voici quelques explications …   1. La première règle autorise tout le trafic sur l'interface de *loopback* locale grâce à -i lo.  Il n'y a pas de risque à autoriser l’hôte à communiquer avec lui-même, d’autant plus qu’il en a parfois besoin. 2. La seconde règle autorise toutes les connexions qui sont déjà à l'état ESTABLISHED ou RELATED.  En clair, elle autorise toutes les connexions qui ont été initiées par l’hôte.  Là encore, cela permet d'assouplir le pare-feu et de le rendre fonctionnel pour une utilisation quotidienne.   **Signification des paramètres de *--state* …**   * NEW (NOUVEAU) Signifie que le paquet a démarré une nouvelle connexion, ou autrement associé à une connexion qui n'a pas vu les paquets dans les deux sens ; * ESTABLISHED (ÉTABLIE) Signifie que le paquet est associé à une connexion qui a vu des paquets dans les deux directions ; * RELATED (RELATIVE) Signifie que le paquet a démarré une nouvelle connexion, mais est associé à une connexion existante, comme un transfert de données FTP, ou une erreur ICMP.   Refuser toutes les autres connexions par défaut  Il reste un point essentiel à traiter car, **pour l'instant, ce filtrage ne sert à rien**.  En effet, il est indiqué quelles données autorisées, mais **on n'a pas dit que toutes les autres devaient être refusées** !  On doit donc modifier la **règle par défaut pour DROP** par exemple :  **#** **iptables -P INPUT DROP**  *iptables* devrait maintenant indiquer que par défaut tout est refusé, sauf ce qui est indiqué par les lignes dans le tableau :  **#** **iptables -L**  Chain INPUT (policy DROP)  target      prot  opt  source destination  ACCEPT      tcp   --   anywhere  anywhere            tcp dpt:www  ACCEPT      tcp   --   anywhere anywhere            tcp dpt:ssh  ACCEPT      tcp   --   anywhere  anywhere            tcp dpt:imap2  ACCEPT      icmp  --   anywhere  anywhere  Le **filtrage est radical**. On n’a pas autorisé beaucoup de ports et il se pourrait que l’on se rende compte que **certaines applications n'arrivent plus à accéder à l'internet** (normal, leur port doit être filtré).  Il est bon de savoir quels ports ces applications utilisent pour modifier les règles en conséquence. Au besoin, il faut penser à ajouter de même pour les règles de sortie (OUTPUT).  Annexe A  Gérer ses règles iptables. Proprement.  On voit régulièrement des administrateurs sauvegarder leurs règles iptables dans un script init. Bien que cela fonctionne, ce n'est pas la manière la plus élégante.  Un alternative est la sauvegarde des règles existantes à l'aide de la commande iptables-save.  Cette commande renvoie le résultat sur la sortie standard, redirigeant donc la sortie vers un fichier.  **#** **iptables-save > /etc/iptables**  Une option intéressante… le -c, qui permet d'afficher combien de fois la règle a été appliquée (en bytes et en paquets).  Pour charger les règles, il suffit d'utiliser la commande iptables-restore en fournissant le fichier de sauvegarde en entrée.  **#** **iptables** **-restore < /etc/iptables**  Ajoutez le -c pour prendre en compte les compteurs (s'ils ont été sauvegardés). Si jamais des règles sont déjà chargées, vous n'avez même pas besoin de purger (flusher), la commande le fait pour vous par défaut (-n pour les garder).  Pour restaurer automatiquement les règles au démarrage, on pourrait créer un petit script init.  Il est préférable de directement lancer la restauration avant que l'interface ne soit active (up). Cela permet d'éviter tout oubli lorsquel’on efface les règles puis l’activation des interfaces.  De plus, on gagne ainsi la possibilité de créer des règles personnalisées pour chaque interface.  Si les règles sont indépendantes des interfaces, on peut toujours toujours lancer la restauration en avant l’activation de la boucle locale.  Fichier */etc/network/interfaces*  **iface eth0 inet static**  **pre-up iptables-restore < /etc/iptables**  **address 543.454.233.42**  **netmask 255.255.255.0**  **gateway 543.454.233.254**  **auto eth0**  Annexe B Tables iptables  Netfilter est constitué de 4 tables qui lui permettent de remplir ses fonctions.  **Table Mangle** La première table (MANGLE) par laquelle passe le paquet permet de lui apporter des modifications, leur appliquant ainsi un traitement spécial.  Ceci peut influencer d’autres règles contenues dans les tables NAT et de filtrage.  Elle contient les cinq chaînes prédéfinies.  **Table Filter** La table de filtrage (FILTER) sert principalement de fonctions de pare-feu.  Elle spécifie les règles de trafic en indiquant par quels ports de l’hôte le paquet peut transiter.  Elle est la table par défaut si la chaine n’est associée à aucune table.  Elle possède trois chaînes prédéfinies …   * **Chaîne FORWARD** Filtre et transfère les paquets destinés à une machine derrière le pare-feu. Donc qui le traversent ; * **Chaîne INPUT** Filtre les paquets destinés au pare-feu ; * **Chaîne OUTPUT** Filtre les paquets qui ont pour origine le pare-feu.   **Table NAT** La deuxième table est celle qui gère la translation d’adresse réseau (NAT).  Elle permet de configurer la gestion de l’adressage lorsqu’on désire utiliser des adresses privées au sein du réseau interne.  Or pour router les paquets de données sur un réseau externe, ceux-ci doivent être dotés d’une adresse publique.  Le nating permet donc d’associer une adresse publique à une adresse privé.  Elle contient les chaînes suivantes …   * **Chaîne PREROUTING** (pré-routage) Principalement utilisé pour le DNAT (destination NAT) dans lequel on modifie l'adresse de destination des paquets provenant du réseau public en une adresse privée du réseau local ; * **Chaîne POSTROUTING** (post-routage)  Chaine utilisée pour le SNAT (source NAT) et qui consiste à modifier une adresse privée source d’un paquet sortant avec une adresse source publique ; * **Chaîne OUTPUT** Chaineutilisée pour traduire les adresses des paquets générés par le pare-feu.   **Table RAW** Cette table est principalement utilisée pour marquer les paquets qui doivent ou non être vérifiés par le système de traçage de connexion.  Celui-ci permet de connaître l’état d’une connexion spécifique et ainsi de contrôler plus facilement qui ou quoi est autorisé à démarrer de nouvelles sessions.  Références  Adapté des textes originaux <http://fr.openclassrooms.com/informatique/cours/reprenez-le-controle-a-l-aide-de-linux/iptables-le-pare-feu-de-reference>  <http://www.planet-libre.org/index.php?post_id=15665&go=external>  <http://doc.ubuntu-fr.org/iptables>  Autres références  <http://wiki.ubuntu.com/IptablesHowTo>  <http://www.inetdoc.net/guides/iptables-tutorial/introduction.html>  ***Iptables* - Résumé des commandes**  **Lister les règles existantes**  **iptables -L**  **+++ afficher les numéros de lignes**  **iptables -L --*line-numbers***  **Effacer les règles existantes**  **iptables –F**  **Ajouter et supprimer des règles**  **Ajouter une règle en fin de liste pour la chaine indiquée (*INPUT* ou *OUTPUT*, par exemple)**  **iptables -A *chaîne***  **Supprime la règle n° rulenum pour la chaine indiquée**  **iptables -D *chaîne rulenum***  **Insérer une règle au milieu de la liste à la position indiquée par rulenum.**  **iptables -I *chaîne rulenum***  **#** Si aucune position *rulenum* n’est indiquée, la règle sera insérée en premier, tout en haut dans la liste.  **Remplacer la règle n° *rulenum* dans la chaine indiquée iptables -R *chaîne rulenum***  **Modifier la règle par défaut pour la chaine**  **iptables -P *chaîne règle***  **L’ajout d'une règle se passe suivant ce schéma:**  **iptables -A (chain) -p (protocole) --dport (port) -j (décision)**  **iptables -A INPUT -p tcp --dport www -j ACCEPT**  **iptables -A INPUT -p tcp --dport imap4 -j ACCEPT**  **iptables -A INPUT -p icmp -j ACCEPT**  **iptables -A INPUT -i lo -j ACCEPT**  **iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT**  **iptables -P INPUT DROP**  **Sauvegarder et restaurer les règles**  **iptables-save > /etc/iptables**  **iptables -restore < /etc/iptables**  Annexe A Script iptables  Ce script est un exemple, il est à adapter selon ses besoins.  Il peut toutefois être utilisé pour une utilisation courante, il offre une plutôt bonne "protection".  Attention…  iptables -A INPUT -m pkttype –pkt-type broadcast -j DROP peut poser problème au DHCP "le broadcast : pour trouver et dialoguer avec un serveur DHCP, la machine va simplement émettre un paquet spécial de broadcast (broadcast sur 255.255.255.255 avec d'autres informations comme le type de requête, les ports de connexion…) sur le réseau local.  Lorsque le serveur DHCP recevra le paquet de broadcast, il renverra un autre paquet de broadcast (n'oubliez pas que le client n'a pas forcement son adresse IP et que donc il n'est pas joignable directement) contenant toutes les informations requises pour le client" cette commande peut donc empêcher le pc de recevoir son ip par le DHCP  **Remarque** … Ce script ne permet pas les connexions à distance. Veuillez ajouter deux règles pour autoriser SSH …  **$IPTS -t filter -A OUTPUT -p tcp --dport 22 -j ACCEPT**  **$IPTS -t filter -A INPUT -p tcp --dport 22 -j ACCEPT**  **Iptables.sh**  **#!/bin/bash**  **IPTS=iptables**  **# Vidage des tables et des règles personnelles**  **$IPTS -t filter -F**  **$IPTS -t filter -X**  **# Interdire toutes connexions entrantes et sortantes**  **$IPTS -t filter -P INPUT DROP**  **$IPTS -t filter -P FORWARD DROP**  **$IPTS -t filter -P OUTPUT DROP**  **# Traitement interface local**  **$IPTS -A INPUT -i lo -j ACCEPT**  **$IPTS -A OUTPUT -o lo -j ACCEPT**  **# Ne pas casser les connexions établies**  **$IPTS -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT**  **$IPTS -A OUTPUT -m state --state RELATED,ESTABLISHED -j ACCEPT**  **# HTTP et HTTPS**  **$IPTS -t filter -A OUTPUT -p tcp --dport 80 -j ACCEPT**  **$IPTS -t filter -A OUTPUT -p tcp --dport 443 -j ACCEPT**  **$IPTS -t filter -A INPUT -p tcp --dport 80 -j ACCEPT**  **$IPTS -t filter -A INPUT -p tcp --dport 443 -j ACCEPT**  **# On rejette les balayages (scans) XMAS et NULL.**  **$IPTS -A INPUT -p tcp --tcp-flags FIN,URG,PSH FIN,URG,PSH -j DROP**  **$IPTS -A INPUT -p tcp --tcp-flags ALL ALL -j DROP**  **$IPTS -A INPUT -p tcp --tcp-flags ALL NONE -j DROP**  **$IPTS -A INPUT -p tcp --tcp-flags SYN,RST SYN,RST -j DROP**  **# Courriels SMTP POP3S (Modifier les ports en fonction de ses besoin)**  **$IPTS -t filter -A INPUT -p tcp --dport 465 -j ACCEPT**  **$IPTS -t filter -A OUTPUT -p tcp --dport 465 -j ACCEPT**  **$IPTS -t filter -A INPUT -p tcp --dport 993 -j ACCEPT**  **$IPTS -t filter -A OUTPUT -p tcp --dport 993 -j ACCEPT**  **# Rejeter silencieusement tous les paquets de diffusion générale (broadcast)**  **# Attention: cfr. message ci-dessus)**  **$IPTS -A INPUT -m pkttype --pkt-type broadcast -j DROP**  **# On journalise les paquets en entrée.**  **$IPTS -A INPUT -j LOG**  **# On journalise les paquets forward.**  **$IPTS -A FORWARD -j LOG**  **exit 0** |
|  |  |  |