

Le firewall du SpeedTouch

dans son intimité

Révision 1.3 du 13 septembre 2004

Le firewall du SpeedTouch dans son intimité

Table des matières

1	Objectif.....	4
2	L'architecture logique.....	5
2.1	Les points d'inspection du firewall.....	5
2.1.1	input.....	5
2.1.2	output.....	6
2.1.3	sink.....	6
2.1.4	source.....	6
2.1.5	forward.....	6
2.2	Les modules d'aiguillage.....	6
2.2.1	Le module routeur (<i>Router module</i>).....	6
2.2.2	Le module de transfert (<i>Forward module</i>).....	6
2.3	Les modules de traitement d'adresses.....	7
2.3.1	Le module NAPT statique (<i>Static NAPT Module</i>).....	7
2.3.2	Le module NAPT dynamique (<i>Dynamic NAPT Module</i>).....	7
3	Le fonctionnement.....	8
3.1	Flux à destination du ST.....	8
3.2	Flux émis par le ST lui-même.....	8
3.3	Flux traversant le ST.....	9
3.4	Qui connaît quoi ?.....	10
4	Les règles firewall.....	11
4.1	Rappels sur les communications TCP/IP.....	11
4.2	Conventions.....	11
4.3	Principes de base.....	12
5	Exemples.....	13
5.1	Les règles protégeant le modem.....	13
5.1.1	Filtre d'accès au modem.....	13
5.1.2	Filtre de sortie du modem.....	14
5.1.3	Pinguer ou non le modem.....	14
5.2	Le firewall niveau 2.....	15
5.2.1	Le filtrage.....	15
5.2.2	Autoriser des accès extérieurs.....	15
5.3	Le firewall niveau 3.....	16
5.3.1	Le filtrage.....	16
5.3.2	Autoriser les flux sortants.....	16
5.3.3	Autoriser les flux entrants.....	17
6	Conclusions.....	18

Le firewall du SpeedTouch dans son intimité

Révision	Date	Commentaires
1.3	13/09/2004	Remise en forme de quelques points (merci <i>Nuts</i>). Note sur les interactions defserver/DynDNS du ST510v3 (merci <i>eblot</i>).
1.2	30/07/2004	Mises à jour concernant les paquets non NATés. Mises à jour concernant les flux du modem lui-même.
1.1	09/07/2004	Corrections cosmétiques et orthographico-grammaticales.
1.0	03/07/2004	Première version publique.

1 Objectif

Le tutoriel de Damocles¹ sur le firewall des SpeedTouch (Pro with firewall, 510v3, 510v4 et dérivés) présente un aperçu rapide du mode de fonctionnement et quelques principes de base du paramétrage, comme la notion de *chaînes*.

Le présent document va s'efforcer de décrire plus précisément cette logique interne, les interactions entre les différents modules (firewall, translation d'adresses (NAT), ...) et les principes généraux servant à construire des règles firewall. Bien sûr, il ne s'agit pas d'un cours réseau, les notions de base du réseau TCP/IP (adresse, port, paquet, ...) sont supposées familières.

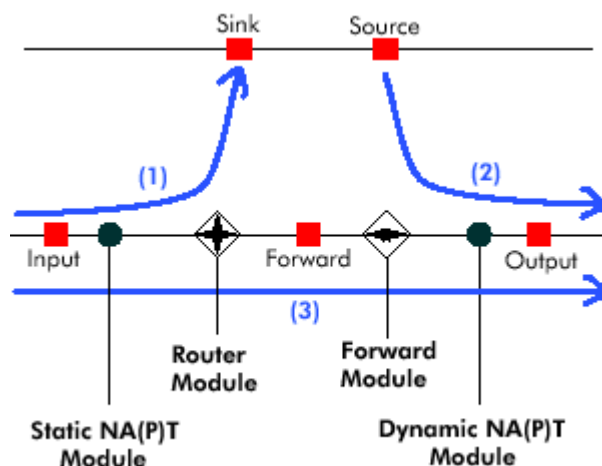
La lecture du tutoriel de Bac's² sur le NAT est également un préalable indispensable.

¹ <http://www.forpage.com/forum/viewtopic.php?t=3343>

² <http://www.forpage.com/forum/viewtopic.php?t=1280>

2 L'architecture logique

Détaillons le (désormais célèbre) schéma de principe du firewall des ST.



2.1 Les points d'inspection du firewall

On pourra trouver également la terminologie de *point d'entrée*, *hook* (crochet) ou *Packet Interception Point* (point d'interception de paquets).

Chaque point d'inspection permet d'observer l'état des paquets TCP/IP. Lorsqu'un paquet passe par un des ces points, **il est dans un état parfaitement défini** : on sait d'où il vient, où il va, quel est le type du protocole IP, si il a subi un traitement sur les adresses et lequel, etc... Il est donc possible de **tester** cet état et d'en **déduire** une action. “**Tester et agir**”, c'est la définition d'une **règle firewall**.

Les règles firewall se définissent sur ces points d'inspection.

IMPORTANT

Ce n'est pas le point d'inspection qui permet de déterminer la provenance d'un paquet.

Les mêmes points d'inspection sont utilisés pour les paquets arrivant d'internet par l'interface WAN et pour les paquets provenant du réseau local par l'interface LAN. C'est uniquement l'examen du paquet à un point d'inspection donné qui permettra de savoir d'où il vient et où il va.

Ces points d'inspection sont au nombre de 5. Ils sont décrits succinctement ci-dessous et le détail des opérations sera examiné au chapitre 3.

2.1.1 input

Par ce point d'inspection, nous allons pouvoir examiner les paquets **qui entrent** dans le ST.

Tous les paquets qui arrivent passent **obligatoirement** par ce point d'observation, qu'ils viennent d'internet (WAN) ou du réseau local (LAN).

Quand ils sont testés dans *input*, les paquets n'ont subi aucun traitement.

Si le paquet provient d'internet, l'adresse destination est l'adresse publique du modem, celle attribuée par le fournisseur d'accès.

Si le paquet arrive du LAN, l'adresse de l'émetteur (adresse source) est l'adresse privée de la machine ayant envoyée le paquet. L'adresse destination est, en général, une adresse internet publique.

La destination locale n'est pas encore connue. Il ne servira donc à rien d'écrire des règles firewall faisant référence à une adresse privée du LAN.

2.1.2 output

Par ce point d'inspection, nous allons pouvoir examiner les paquets **qui sortent** du ST. **Tous** les paquets sortant passent **obligatoirement** par ce point d'observation, qu'ils aillent vers internet ou vers le réseau local.

Quand ils sont testés dans **output**, les paquets ont subi tous les traitements possibles.

Si le paquet provient du réseau local et doit aller vers internet, l'adresse (privée) de l'émetteur a été translatée (NAT) en adresse publique (celle du modem).

Si le paquet arrive d'internet vers le LAN, l'adresse destinataire est maintenant l'adresse privée de la machine visée sur le réseau local.

2.1.3 sink

Par ce point d'inspection, nous allons pouvoir examiner les paquets qui sont entrés et qui sont **destinés au ST lui-même**, explicitement (communication depuis le LAN) ou par défaut (absence de NAT). Les paquets qui peuvent être examinés dans **sink** sont déjà passés par **input** et par le *module NAPT statique*. Nous verrons plus loin tout ce que cela implique (§3.1).

2.1.4 source

Par ce point d'inspection, nous allons pouvoir examiner les paquets **émis par le ST lui-même** pour ses besoins propres. Les paquets observés ici n'ont subi aucun traitement. L'adresse source est l'adresse du ST (locale ou WAN suivant la destination du paquet).

2.1.5 forward

C'est un point d'observation un peu particulier. On peut y tester les paquets qui doivent **traverser** le modem, que ce soit pour aller du réseau local vers internet ou d'internet vers le réseau local. Les paquets observés ont déjà subi quelques transformations, mais pas toutes.

En particulier, l'adresse destinataire est bien l'adresse définitive (utile si le paquet doit aller vers une machine du réseau local). L'adresse de l'émetteur est encore l'adresse réelle.

2.2 Les modules d'aiguillage

Ces modules, non paramétrables, vont aiguiller les paquets au sein du ST, en fonction de diverses caractéristiques, comme l'adresse de destination.

2.2.1 Le module routeur (*Router module*)

Module stratégique, puisque c'est ici que le ST décidera **automatiquement** si le paquet qui vient d'arriver et qui a été accepté est destiné au ST lui-même ou doit traverser le ST.

2.2.2 Le module de transfert (*Forward module*)

Ce module se charge de véhiculer les paquets provenant d'origines diverses (ST ou en transit) vers la sortie.

2.3 Les modules de traitement d'adresses

Ces modules NAPT (*Network Address and Port Translation* – translation d'adresse et de port réseau) sont le coeur du fonctionnement des ST, et, plus généralement, de tous les systèmes qui offrent une fonctionnalité NAT.

Le fonctionnement du NAT est expliqué dans le tutoriel de Bac's³. L'important est de retenir que ces modules NAPT se partagent la même table de travail (table NAT), qui contient donc la “mémoire” des translations d'adresses/ports effectuées.

Il convient également de noter que le comportement de ces modules est influencé par l'utilisation des *helpers* (modules d'applications) dans la commande `nat bind`. Mais c'est une autre histoire !

2.3.1 Le module NAPT statique (*Static NAPT Module*)

Ce module transforme (translate ou convertit) les adresses de **destination** entre adresses privée et publique (privée vers publique ou publique vers privée, suivant le sens du trafic).

Le comportement de ce module est en partie géré par les *règles NAT* du ST (`:nat create protocol=...`). Ces règles NAT permettent de “pré-remplir” la table NAT avec une association entre (`@IP_externe:port_externe`) et (`@IP_ServeurLocal:port_ServeurLocal`).

2.3.2 Le module NAPT dynamique (*Dynamic NAPT Module*)

C'est ce module qui transforme (translate ou convertit) les adresses **source** (émetteur) entre adresses privée et publique. Ce module n'est pas paramétrable. Aucune commande du ST ne permet d'en modifier directement le comportement.

³ <http://www.forpage.com/forum/viewtopic.php?t=1280>

3 Le fonctionnement

Le plus simple pour comprendre le fonctionnement logique du ST est de suivre le cheminement des 3 flux susceptibles d'être traités par le ST. Sur le schéma, ce sont les flèches bleues.

Il faut à nouveau rappeler que ces schémas ne présument pas de l'origine du flux. Les paquets en provenance du LAN ou du WAN sont traités de la même façon.

Notons également que lorsque le ST est connecté sur internet, il possède deux adresses IP: une adresse locale (10.0.0.138 par défaut) et une adresse publique internet (adresse WAN) attribuée par le fournisseur d'accès.

3.1 Flux à destination du ST

On trouve dans ce type de flux (*flux 1*), par exemple, les commandes telnet (administration depuis le LAN) ou les réponses des serveurs DNS sur requête du ST, mais aussi les retours liés aux requêtes DNS dynamiques et mail automatique des dernières versions du firmware du ST510v3.

Plus important, les paquets en provenance du WAN sans correspondance dans la table NAT suivent naturellement ce chemin.

1. Comme tout ce qui rentre dans le ST, le paquet traverse donc d'abord **input**. Si le paquet provient d'internet, l'adresse destination est l'adresse publique du ST.
2. Si aucune règle ne bloque le paquet dans **input**, le paquet passe dans le **module NAPT statique**. Le module consulte la table NAT et détermine ainsi quelle est l'adresse réelle que doit contenir le paquet.

Si le paquet provient du réseau local, l'adresse destination est déjà l'adresse réelle locale, il n'y a donc pas de transformation.

Tous les paquets qui arrivent du monde du dehors ont pour adresse de destination l'adresse WAN du ST (IP publique du FAI). La table NAT devrait permettre de retrouver à quelle station (adresse) du LAN ce paquet est destiné en fonction des ports utilisés.

Si aucune règle NAT ne peut prendre en charge le paquet en cours, le **module NAPT statique** ne change rien aux informations d'adressage. L'adresse de destination du paquet reste l'adresse réelle WAN du routeur.

3. Le paquet ainsi traité (et non transformé) passe alors par le **module routeur**. L'adresse de destination est bien l'adresse du routeur (locale ou WAN), le paquet est donc aiguillé vers le ST lui-même au travers du point d'entrée **sink**.
4. Le paquet à destination du modem peut désormais être testé pour action grâce aux règles définies dans le point d'entrée **sink**.

Cette description permet de comprendre qu'un paquet ne peut arriver au modem sur un port précis que si les conditions suivantes sont réunies:

- Aucune règle définie en **input** n'empêche le paquet d'entrer.
- Aucune règle NAT ne modifie la destination du paquet sur le port souhaité.

Ceci explique pourquoi l'utilisation des fonctionnalités de mise à jour DynDNS ou de mail automatique du ST510v3 n'est pas compatible avec une règle NAT *defserver*. Cela sort du cadre de ce tutoriel, mais vous trouverez un palliatif sur Forpage⁴, en attendant un nouveau firmware.

3.2 Flux émis par le ST lui-même

Ce *flux 2* concerne, par exemple, les réponses aux commandes d'administration (telnet, FTP, Web), les requêtes DNS émises par le relais interne du ST ou les mises à jour DynDNS.

Le ST ayant 2 interfaces réseau (une vers le réseau local et une vers internet), il choisira de faire partir les paquets par celle qui est connectée à la destination. L'adresse source du paquet émis reflètera ce choix.

⁴ <http://www.forpage.com/forum/viewtopic.php?p=59379#59379>

1. Le paquet visible par le point d'entrée **source** a les caractéristiques suivantes:
 - l'adresse émetteur (source) est l'adresse locale du ST si le paquet est à destination du LAN ou l'adresse WAN du ST si le paquet est à destination d'internet.
 - l'adresse destination est l'adresse réelle (privée ou publique).
2. Si le paquet a passé le filtre **source**, il va arriver dans le *module de transfert* qui va le diriger vers le **module NAPT dynamique**.
3. Ce module **module NAPT dynamique** "gère" l'adresse source du paquet. Il doit faire en sorte que cette adresse soit cohérente avec le réseau vers lequel le paquet doit aller.

En fait, le ST a déjà mis la bonne adresse source (locale ou publique) dans le paquet. Il n'y a donc rien à faire.
4. Le paquet arrive alors dans **output**. Il a les caractéristiques suivantes:
 - l'adresse de destination est l'adresse réelle.
 - l'adresse source est l'adresse utilisable sur le réseau cible: adresse privée si le paquet va sur le réseau local, ou adresse publique si le paquet part sur internet.

Le paquet est prêt à sortir, si il ne trouve pas une règle en **output** qui l'en empêche.

3.3 Flux traversant le ST

Le *flux 3* représente le fonctionnement le plus courant du ST.

1. Comme tout ce qui rentre dans le ST, le paquet traverse donc d'abord **input**. Si le paquet provient d'internet, l'adresse destination est l'adresse publique du ST.
2. Si aucune règle ne bloque le paquet dans **input**, le paquet passe dans le **module NAPT statique**. Le module consulte la table NAT et détermine ainsi quelle est l'adresse réelle que doit contenir le paquet.

Si le paquet provient du réseau local, l'adresse destination est déjà l'adresse réelle, il n'y a donc pas de transformation.

Si le paquet provient du monde du dehors, la table NAT va permettre de retrouver à quelle station (adresse) du LAN ce paquet est destiné en fonction des ports utilisés. Cette information a été créée dynamiquement par le **module NAPT dynamique**, ou statiquement, grâce aux règles NAT.

Le **module NAPT dynamique** met donc à jour le paquet en modifiant l'adresse destination.
3. Le paquet ainsi transformé passe alors par le *module routeur*. L'adresse de destination n'est pas celle du routeur, le paquet est donc aiguillé pour traverser le ST.
4. Le paquet qui passe par le point d'entrée **forward** a donc les caractéristiques suivantes:
 - l'adresse de destination est l'adresse réelle (adresse privée si la destination est une machine du LAN ou adresse internet si la destination est dans le monde du dehors)
 - l'adresse source (émetteur) est l'adresse réelle. Si le paquet provient du LAN, on trouve l'adresse privée de la machine ayant envoyé le paquet.

forward est donc le premier point d'entrée traversé par lequel on peut établir un filtre sur la destination **réelle** du paquet, quelle que soit sa destination (locale ou internet).
5. Si aucune règle n'a bloqué le paquet au point d'entrée **forward**, le ST achemine ce paquet via le *module de transfert* et le fait passer dans le **module NAPT dynamique**. Ce module "gère" l'adresse source du paquet. Il doit faire en sorte que cette adresse soit cohérente avec le réseau vers lequel le paquet doit aller.

Si le paquet doit aller sur internet, il transforme l'adresse privée de l'émetteur en une adresse publique. Et, pour ne pas perdre le fil des échanges, ce module met également à jour la table NAT en mémorisant les informations (adresses réelles, port source, port destination).

Si le paquet est à destination du réseau local, l'adresse source n'est pas modifiée.
6. Le paquet arrive alors dans **output**. Il a les caractéristiques suivantes:
 - l'adresse de destination est l'adresse réelle
 - l'adresse source est l'adresse publique du ST si le paquet part vers internet ou l'adresse réelle si le paquet va sur le réseau local.

3.4 Qui connaît quoi ?

Le tableau suivant permet de déterminer sur quel type d'adresse chaque point d'entrée du firewall peut travailler. Ici, le terme "adresse" est pris au sens large : adresse IP et port.

Les types d'adresse possibles sont:

- FAI adresse publique WAN du modem, attribuée par le FAI.
- LOC adresse IP privée du réseau local.
- IP adresse IP internet.

		input	sink	forward	source	output
LAN->WAN	Adresse source	LOC	-	LOC	-	FAI (NAT)
	Adresse destination	IP	-	IP	-	IP
WAN->LAN	Adresse source	IP	-	IP	-	IP
	Adresse destination	FAI	-	LOC (NAT)	-	LOC (NAT)
Modem->LAN	Adresse source	-	-	-	LOC	LOC
	Adresse destination	-	-	-	LOC	LOC
LAN->Modem	Adresse source	LOC	LOC	-	-	-
	Adresse destination	LOC	LOC	-	-	-
Modem->WAN	Adresse source	-	-	-	FAI	FAI
	Adresse destination	-	-	-	IP	IP
WAN->Modem	Adresse source	IP	IP	-	-	-
	Adresse destination	FAI	FAI	-	-	-

- Les paquets internet sans correspondance dans la table NAT sont considérés comme étant destinés au ST lui-même (adresse destination = adresse publique du modem).
- Les paquets en provenance du ST lui-même ne sont pas NATés, puisque l'adresse source est déjà au bon format (adresse privée ou publique suivant la destination).

4 Les règles firewall

4.1 Rappels sur les communications TCP/IP

Sans entrer dans les détails, il est nécessaire de préciser quelques notions sur les échanges en TCP/IP. C'est indispensable pour comprendre ensuite la construction des règles firewall.

Le but n'étant pas de faire un cours réseau approfondi⁵, il suffit de retenir de manière **simpliste** les points suivants.

Un *paquet* est l'élément de base. Il contient les données et un en-tête comprenant, entre autres, (adresse:port) **source** et **destination**. On y trouve également des indicateurs utilisés pour contrôler la communication (établissement, fin, acquittement, ...)

Une *communication* se fait sur le mode **client-serveur**. C'est un **échange** de *paquets* entre (*adresse_client:port_client*) et (*adresse_serveur:port_serveur*). Adresses et ports ne changent pas durant **toute** la communication.

Le port utilisé par le client (*port_client*) est choisi **aléatoirement** par le système d'exploitation.

Le port utilisé par le serveur (*port_serveur*) est fonction du service offert. Un serveur telnet sera appelé (écoute) sur le port 23. Un eMule/edonkey écoute sur le port 4662 (le bourricot qui galope à la maison, pas le serveur Razorback).

port_client et *port_serveur* sont **alternativement** port **source** et port **destinataire** dans les en-têtes des paquets échangés, suivant le sens observé. Il faut donc bien dissocier les notions de ports client/serveur (communication) et ports source/destination (échange des paquets).

4.2 Conventions

Comme expliqué dans le tutoriel de Damocles⁶, la mise en place des règles firewall se fait en 3 temps :

- Création des chaînes de règles (*chain create*)
`:firewall chain create chain=INPUT ...`
- Assignment d'une chaîne de règles à chaque point d'entrée (*assign hook*)
`:firewall assign hook=input chain=INPUT ...`
- Création des règles pour chacune des chaînes (*rule create*)
`:firewall rule create chain=INPUT ...`

Les **conventions** adoptées sont les suivantes :

- Il y a autant de chaînes de règles que de points d'entrées (5).
- Le nom de la chaîne correspond au point d'entrée auquel cette chaîne est destinée.
- Le nom de la chaîne est en MAJUSCULES.
- Le nom du point d'entrée est en minuscules.
- Une chaîne de règles peut être vide.

Il s'agit bien de *conventions*. Il n'y a pas d'obligation technique à nommer *INPUT* la chaîne destinée au point d'entrée **input**. Elle peut aussi s'appeler *input*, *ENTREE* ou *TocToc* (entrez !).

⁵ <http://publib-b.boulder.ibm.com/redbooks.nsf/RedbookAbstracts/gg243376.html?Open>

⁶ <http://www.forpage.com/forum/viewtopic.php?t=3343>

Pour être rigoureux, une chaîne de règles est un objet à part entière. Elle peut avoir un nom quelconque, qu'on peut choisir d'après le service qu'elle va contrôler (chaîne *FTP* par exemple), le point d'interception auquel elle va être rattachée (*INPUT*, *FORWARD*, ...) ou même les vainqueurs de la Star Ac'.

L'objet "chaîne de règles" est indépendant des points d'observation. L'étude des possibilités de "programmation" de ces règles sort du cadre de ce tutoriel.

Le nom de la chaîne est sensible à la casse: *INPUT* n'est pas *Input*.

4.3 Principes de base

- Un point d'inspection sans règle (chaîne de règles vide) **ne bloque aucun paquet**.
- Les paquets sont **implicitement acceptés**.
- Les paquets doivent être **explicitement bloqués**.
- Les règles sont traitées en **séquence**, dans l'ordre défini par l'**index**.
- Le traitement d'un paquet dans un point d'inspection s'arrête
 - à la première règle qui vérifie l'état du paquet
 - lorsque toutes les règles ont été testées

Le firewall du SpeedTouch fonctionne donc sur le principe du
Tout ce qui n'est pas explicitement interdit est autorisé.

La configuration par défaut du firewall des SpeedTouch (après un RESET par exemple) protège le modem et laisse passer tous les paquets WAN<->LAN. Les connexions entrantes sont acceptées par le firewall, mais "filtrées" (autorisées/refusées) par la présence ou non de règles NAT.

5 Exemples

La syntaxe complète des commandes de création des règles se trouve dans le Guide des commandes en Ligne⁷ (CLI). Les paramètres sont nombreux, certains seront abordés dans ce qui suit.

Ces quelques exemples vont permettre de voir par la pratique les éléments théoriques développés ci-dessus.

Il ne sera abordé que l'aspect *firewall*. Dans une configuration d'hébergement de serveurs (FTP, Web, P2P, ...), les règles firewall ne suffisent pas, il faut également penser à créer les règles NAT nécessaire à l'établissement de la communication.

Il peut y avoir plusieurs façons d'enchaîner des règles firewall pour atteindre le résultat souhaité.

5.1 Les règles protégeant le modem

Décortiquons les règles installées par défaut sur tous les niveaux de firewall.

5.1.1 Filtre d'accès au modem

Commençons par les règles filtrant les l'accès qui **arrivent** sur le modem (point d'inspection *sink*). Pour rappel, quand ils sont testés dans *sink*, les paquets sont déjà passés par *input*.

sink vérifiant les paquets **destinés** au modem, la plupart des tests se feront en testant le **port de destination** (*dstport*) indiqué dans le paquet. Il correspond au *port_serveur* qu'on cherche à atteindre sur le modem.

La chaîne de règle est construite dans l'idée d'**ignorer** les paquets qui ne sont pas **explicitement** acceptés. C'est le rôle de la dernière règle de la chaîne.

```
:firewall rule create chain=SINK index=0 prot=udp dstport=dns action=accept
```

Le ST accepte tous les paquets UDP à destination du port (*dstport*) 53 (port DNS), quelle que soit la provenance du paquet (LAN ou WAN). Cette règle autorise le ST à accepter les requêtes DNS, mais également les réponses DNS et permet donc de l'utiliser comme serveur et relais DNS.

```
:firewall rule create chain=SINK index=1 prot=icmp icmptype=echo-request action=accept
```

Le ST accepte tous les paquets ICMP de type *echo-request*, quelle que soit la provenance du paquet (LAN ou WAN). Avec cette règle, le ST accepte d'être "pingué", aussi bien depuis le réseau local que depuis internet.

```
:firewall rule create chain=SINK index=2 srcintf=!eth0 action=drop
```

A partir de cette règle, le ST **ignore** (*drop*) tous les paquets **qui ne proviennent pas** (*srcintf*) du port ethernet LAN, quel que soit le protocole (TCP, UDP, ...). Le caractère "!" (*not*) permet d'inverser le test.

```
:firewall rule create chain=SINK index=3 prot=udp dstport=67 dstportend=68 action=accept
```

Le ST accepte tous les paquets UDP à destination des ports compris (*dstport* et *dstportend*) entre 67 et 68. Il s'agit des ports utilisés par le protocole DHCP.

Il faut noter qu'il n'y a pas de contraintes concernant la provenance (*srcintf*) WAN ou LAN, mais la règle précédente (*index=2*) bloque les appels extérieurs.

⁷ http://www.speedtouchdsl.com/pdf/cli_guide_prowf.pdf
<http://www.speedtouch.com/support.htm>

```
:firewall rule create chain=SINK index=4 prot=tcp srcintf=eth0 dstport=20 action=accept
:firewall rule create chain=SINK index=5 prot=tcp srcintf=eth0 dstport=21 action=accept
:firewall rule create chain=SINK index=6 prot=tcp srcintf=eth0 dstport=23 action=accept
:firewall rule create chain=SINK index=7 prot=tcp srcintf=eth0 dstport=80 action=accept
```

Ces règles autorisent l'accès au modem en FTP (ports 20 et 21), en telnet (port 23) et en HTTP (port 80).

Il faut noter la contrainte concernant la provenance LAN (*srcintf=eth0*) du paquet. Elle fait double emploi avec la règle 2, mais elle rend la règle plus lisible, sans nuire à la performance.

```
:firewall rule create chain=SINK index=8 action=drop
```

Si un paquet n'a pas déjà été intercepté par une des règles précédentes, il est définitivement éliminé par cette règle.

5.1.2 Filtre de sortie du modem

Ces règles vont permettre de définir ce que le modem lui-même est autorisé à faire. Le principe est ici aussi d'**ignorer** tout ce qui n'est pas **explicitement** autorisé.

```
:firewall rule create chain=SOURCE index=0 prot=udp dstport=dns action=accept
```

Le ST peut envoyer des paquets UDP à destination du port (*dstport*) 53 (port DNS), quelle que soit la destination du paquet (LAN ou WAN). Cette règle permet donc au ST de répondre à des requêtes (émises par une station du LAN) ou d'émettre une requête DNS vers un serveur DNS internet.

```
:firewall rule create chain=SOURCE index=1 dstintf=eth0 action=accept
```

Le ST est autorisé à envoyer (*dstintf*) les paquets **uniquement** sur l'interface LAN, quel que soit le protocole. Toutes les réponses aux demandes telnet, FTP ou Web sont prises en compte par cette règle.

```
:firewall rule create chain=SOURCE index=2 prot=udp dstport=67 action=accept
```

Les réponses DHCP sont autorisées, quelle que soit la destination. Pour les réponses vers le LAN, cette règle est redondante avec la règle 1.

```
:firewall rule create chain=SOURCE index=3 prot=icmp icmptype=echo-reply action=accept
```

Le ST est autorisé à envoyer un *echo-reply* aussi bien sur le LAN que sur le WAN. La règle 1 de *sink* acceptait les "pings", cette règle permet d'envoyer la réponse (le "pong").

```
:firewall rule create chain=SOURCE index=4 action=drop
```

Tout ce qui n'entre pas dans les catégories testées par les règles précédentes est ignoré.

5.1.3 Pinguer ou non le modem

Avec les règles présentées ci-dessus, le modem est *pinguable* aussi bien du réseau local que d'internet (par l'adresse WAN). Le *ping* fonctionne sur 2 types de paquet ICMP: *echo-request* est la demande, *echo-reply* est la réponse. Si un seul des paquets est interdit sur la machine cible, le ping ne fonctionnera pas.

Pour supprimer cette fonctionnalité, il suffit donc d'agir sur l'une des règles *SINK/1* ou *SOURCE/3*, ou les deux.

Par ailleurs, nous avons vu que la construction des règles autorise tout le trafic "modem<>LAN" et supprime le trafic "modem<>WAN" non explicitement autorisé.

Pour empêcher le modem d'être "pingué" depuis l'extérieur, il suffit donc de supprimer les règles ICMP. Simple, non ?

5.2 Le firewall niveau 2

Outre les règles protégeant le modem, on trouve également des règles traitant le trafic traditionnel LAN<>WAN. Compte tenu du fonctionnement interne du firewall ST, ces règles sont essentiellement regroupées sur le point d'inspection **input** (chaîne *INPUT*).

5.2.1 Le filtrage

Dans cette configuration, les **demandes de connexion** en provenance de l'extérieur sont interdites. Le trafic LAN>WAN est autorisé, quel qu'il soit, ainsi que le trafic de réponse WAN>LAN.

```
:firewall rule create chain=INPUT index=0 srcintfgrp=wan prot=tcp syn=yes ack=no action=drop
```

Cette règle vérifie les paquets en provenance d'internet (*srcintfgrp=wan*) et contrôle 2 indicateurs (*SYN* et *ACK*) présents dans l'en-tête. Les demandes de connexion se caractérisent toujours par ces valeurs *SYN=YES ACK=NO*. Ces requêtes sont donc ignorées, empêchant quiconque de se connecter depuis l'extérieur.

```
:firewall rule create chain=INPUT index=1 srcintfgrp=wan prot=udp dstport=135 dstportend=139 action=drop
:firewall rule create chain=INPUT index=2 srcintfgrp=wan prot=tcp dstport=135 dstportend=139 action=drop
:firewall rule create chain=INPUT index=3 srcintfgrp=wan prot=tcp dstport=593 action=drop
:firewall rule create chain=INPUT index=4 srcintfgrp=wan prot=tcp dstport=445 action=drop
```

Ces règles permettent de bloquer les méchancetés virales générées par quelques infections microsoftiennes.

Il est important de noter qu'il n'y a pas de règle "par défaut" qui jette tout le trafic inconnu.

5.2.2 Autoriser des accès extérieurs

Le niveau 2 empêchant les connexions en provenance de l'extérieur, il est donc impossible, en l'état, d'héberger un serveur sur son LAN, que ce soit un serveur Web, FTP ou P2P. Il faut rajouter des règles qui vont autoriser **explicitement** ces demandes d'accès.

L'évaluation des règles se fait dans l'ordre des index. Ces nouvelles règles doivent donc se placer **avant** celle bloquant les accès.

Par exemple, voici les règles *INPUT* à utiliser pour avoir un statut HighID sur le réseau eMule/edonkey. Ce statut ne peut s'obtenir que si le port 4662 (par défaut) du client installé sur le PC est accessible depuis l'extérieur (l'extérieur peut se **connecter**). Une règle NAT est bien sûr indispensable, comme pour tout hébergement de serveur, mais il faut également autoriser cette connexion extérieure. Elle peut venir de partout, mais elle visera toujours le port 4662. On obtient donc les règles suivantes:

```
:firewall rule create chain=INPUT index=0 srcintfgrp=wan prot=tcp dstport=4662 action=accept
:firewall rule create chain=INPUT index=1 srcintfgrp=wan prot=tcp syn=yes ack=no action=drop
:firewall rule create chain=INPUT index=2 srcintfgrp=wan prot=udp dstport=135 dstportend=139 action=drop
:firewall rule create chain=INPUT index=3 srcintfgrp=wan prot=tcp dstport=135 dstportend=139 action=drop
:firewall rule create chain=INPUT index=4 srcintfgrp=wan prot=tcp dstport=593 action=drop
:firewall rule create chain=INPUT index=5 srcintfgrp=wan prot=tcp dstport=445 action=drop
```

5.3 Le firewall niveau 3

Ce mode est le plus restrictif, puisqu'il **bloque** tout le trafic **non explicitement autorisé**.

5.3.1 Le filtrage

C'est simple: tout, ou presque, est bloqué !

```
:firewall rule create chain=INPUT index=0 srcintfgrp=lan dstintfgrp=lan action=accept
```

Cette règle peut paraître surprenante, puisque le trafic LAN<>LAN n'est pas supposé passer par le routeur. Certes ! Mais n'oublions pas que le modem, qui est bien sur le LAN, doit rester accessible ! Il faut donc bien autoriser explicitement ces flux.

De plus, le ST peut gérer plusieurs adresses IP sur son interface LAN. Cette règle permet donc en plus de continuer à faire dialoguer les différents réseaux raccordés sur cette interface.

```
:firewall rule create chain=INPUT index=1 srcintfgrp=lan dstintfgrp=wan action=drop
:firewall rule create chain=INPUT index=2 srcintfgrp=wan dstintfgrp=lan action=drop
```

Est-il besoin de commenter ? Le trafic LAN>WAN (règle 1) est interdit, ainsi que le trafic WAN>LAN (règle 2), quel que soit le protocole.

Pour optimiser les traitements, ces deux règles peuvent d'ailleurs se combiner en une seule:

```
:firewall rule create chain=INPUT index=1 action=drop
```

5.3.2 Autoriser les flux sortants

Un modem-routeur qui bloque tout, ce n'est pas ce qu'il y a de plus utile. Il faut donc ajouter les règles qui vont permettre d'utiliser le réseau. Comme tout le trafic est interdit par défaut, ces nouvelles règles devront donc autoriser explicitement les **deux sens** du flux.

C'est ici qu'interviennent les notions importantes et différentes de *port_serveur/port_client* (communication) et de *port_source/port_destination* (paquet).

Commençons par autoriser les postes du LAN à surfer.

Les serveurs Web sont appelés sur le port 80 (*port_serveur*). Quand le paquet part du client vers le serveur, son *port_destination* est bien 80, et son *port_source* a une valeur choisie par le système.

Inversement, quand le serveur Web répond à la requête du client, *port_serveur* est toujours 80 (c'est toujours un serveur Web). Mais, dans le paquet qui va du serveur vers le client, c'est désormais *port_source* qui vaut 80. Le champ *port_destination* contient la valeur du port que le client a utilisé pour se connecter.

Ce qui nous donne donc les règles 1 et 2 supplémentaires suivantes:

```
:firewall rule create chain=INPUT index=0 srcintfgrp=lan dstintfgrp=lan action=accept
:firewall rule create chain=INPUT index=1 prot=tcp dstintfgrp=wan dstport=80 action=accept
:firewall rule create chain=INPUT index=2 prot=tcp dstintfgrp=lan srcport=80 action=accept
```

La règle 1 autorise les paquets à destination du WAN (*dstintfgrp=wan*) si ils vont vers le port HTTP (*dstport=80*).

La règle 2 autorise les paquets en sens inverse, à destination du LAN (*dstintfgrp=lan*) et **provenant** d'un serveur web (*srcport=80*). Cette règle aurait également pu s'écrire en précisant que la source était le WAN (*srcintfgrp=wan*), le résultat est identique.

Ne pas oublier de terminer la chaîne en ignorant tous les autres paquets non désirés.

```
:firewall rule create chain=INPUT index=3 action=drop
```

5.3.3 Autoriser les flux entrants

Le principe est identique, à ceci près que *port_serveur* est maintenant un port sur le réseau local. Il faudra donc également penser à établir la règle NAT qui convient.

Prenons l'exemple (assez complexe) d'un client eMule installé sur un des postes du réseau. Dans le système P2P, chaque eMule est à la fois client d'autres sources (qui écoutent sur le port 4662 ou autres) et serveur (à l'écoute du port 4662 par défaut). L'aide en ligne d'eMule fournit la liste de tous les ports utilisés pour communiquer, ainsi que le sens à prendre en compte.

Le client et les serveurs eMule ont des paramètres réseau très facilement modifiables. En particulier, l'utilisation des ports par défaut est de moins en moins courante. Il est donc très restrictif de supposer que le client eMule du réseau n'a besoin de se connecter que sur le port 4662 des autres sources. Mais il est également difficile d'ouvrir tous les ports, sinon l'intérêt du firewall niveau 3 disparaît !

Commençons par permettre au client eMule de se connecter à d'autres sources utilisant le port 4662. Il faut bien sûr prévoir également le flux de retour, partant du port 4662 de la source contactée.

```
:firewall rule create chain=INPUT index=0 srcintfgrp=lan dstintfgrp=lan action=accept
:firewall rule create chain=INPUT index=1 prot=tcp dstintfgrp=wan dstport=4662 action=accept
:firewall rule create chain=INPUT index=2 prot=tcp dstintfgrp=lan srcport=4662 action=accept
```

Ensuite, ce client eMule doit également agir comme source. Il doit donc être joint sur son port 4662 (règle NAT à prévoir en plus), et il doit pouvoir répondre depuis ce port.

```
:firewall rule create chain=INPUT index=3 prot=tcp srcintfgrp=wan dstport=4662 action=accept
:firewall rule create chain=INPUT index=4 prot=tcp srcintfgrp=lan srcport=4662 action=accept
```

A titre d'exercice, il ne reste plus qu'à relire la documentation d'eMule et écrire les règles concernant chaque port utilisé par le bourricot.

Ce simple exemple, pourtant très courant, montre bien la difficulté qu'il y a à utiliser le firewall niveau 3 pour certains services. En particulier, **le niveau 3 n'est pas adapté pour un fonctionnement efficace des logiciels de Peer to Peer**, comme eMule/edonkey.

6 Conclusions

Les options de création des règles sont très nombreuses et l'écriture des chaînes peut se révéler très vite un exercice particulièrement complexe. Comme souvent, il est préférable de privilégier la simplicité pour garder le maximum de souplesse. Un firewall niveau 1 et quelques règles NAT seront plus efficaces et performants qu'un firewall niveau 3 mal maîtrisé.

Le firewall du SpeedTouch ne sait pas tout faire ! Il protège très bien le réseau local des tentatives d'intrusion. Par contre, il est incapable de dire si un flux autorisé est réellement légitime. Si le port 80 est autorisé en sortie, tout ce qui emprunte ce chemin sera autorisé, que le flux provienne d'un navigateur ou d'un Cheval de Troie.

Ceux que la sécurité titille associent souvent la protection entrante apportée par le firewall du ST à un firewall logiciel installé sur tous les postes du LAN. Ce firewall logiciel permet de contrôler les flux par application, et non plus par la seule utilisation d'un port.