Filtrage IP avec Shorewall, une solution légère

Stéphane Bortzmeyer

<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 10 avril 2007. Dernière mise à jour le 19 décembre 2011

<https://www.bortzmeyer.org/filtrage-avec-shorewall.html>

Aujourd'hui, il n'est plus pensable de connecter un serveur à Internet sans qu'il soit protégé par un coupe-feu. Shorewall est un moyen assez simple de configurer un coupe-feu sur Linux.

Il existe de nombreuses sortes de coupe-feu et de nombreuses façons de les configurer. Par exemple, sur Linux, le coupe-feu de base est Netfilter et se programme avec des commandes shell (`iptables`), pour lesquelles on dispose de toute la puissance de ce langage.

Mais ce système a des limites. Par exemple, certains souhaitent une interface graphique plus facile à utiliser et il en existe beaucoup. D'autres ont des routeurs de plusieurs modèles à gérer et ils souhaitent un langage qu'on puisse traduire automatiquement en Netfilter, IOS ou autre selon le modèle de routeur (actuellement, seul `hfl` <<http://www.hfl1.org/>> est sur ce créneau). D'autres ont un réseau un peu complexe, avec plusieurs points d'entrée, et souhaitent centraliser la politique de filtrage en un seul groupe de fichier, un programme de routage produisant ensuite les règles de filtrage pour chaque machine (seul `netspoc` <<http://netspoc.berlios.de/>> occupe ce créneau). Le projet **genice**, de Vincent Archer, visait à répondre à ces deux dernières demandes mais n'a hélas pas débouché. J'ai gardé le code, si quelqu'un est intéressé.

Ma demande était plus modeste : protéger une machine Linux, un serveur connecté à Internet derrière un routeur que je ne contrôle pas et sur lequel je ne pouvais donc pas compter. J'aurai pu faire du Netfilter à la main mais c'était l'occasion d'utiliser Shorewall et c'est en effet très simple. Avec seulement la configuration suivante (largement copiée du fichier `Samples/one-interface` livré avec Shorewall), j'ai un coupe-feu filtrant qui marche.

```
# Zones :
# On définit les zones entre lesquelles on filtre. Ici, le coupe-feu
# lui-même et l'Internet, tout le reste.
fw      firewall
net     ipv4

# Interfaces :
```

```

# Quelle interface connecte les deux zones ? Ici, eth0. Notez les
# options à la fin.
net      eth0          detect          norfc1918,routefilter,tcpflags,logmartians,nosmurfs

# Policy:
# Quelle politique par défaut. Notez le $FW, qui référence la zone
# configurée plus tôt. Ici, par défaut, tout ce qui vient du coupe-feu
# est accepté, tout ce qui y arrive est refusé.
$FW      net          ACCEPT
net      $FW         DROP          info
net      all         DROP          info
all      all         REJECT       info

# Rules:
# Les exceptions à la politique par défaut. Le serveur/coupe-feu étant
# un serveur HTTP, DNS et SMTP, on autorise ces protocoles.
# SSH
ACCEPT          net      fw      tcp      22
# SMTP
ACCEPT          net      fw      tcp      25
# Web
ACCEPT          net      fw      tcp      80
ACCEPT          net      fw      tcp      443
# DNS
ACCEPT          net      fw      tcp      53
ACCEPT          net      fw      udp      53
# ping
ACCEPT          net      fw      icmp     8

```

(Note sur la syntaxe : on peut aussi utiliser des macros Shorewall qui permettent de simplifier certaines définitions. Ainsi, `Web/ACCEPT net fw` utilise la macro `Web` qui met automatiquement deux règles, pour les ports 80 et 443.)

Le premier méchant attrapé a été enregistré :

```

Apr 10 16:09:04 munzer Shorewall:net2fw:DROP:IN=eth0 OUT=
MAC=00:16:3e:29:6c:26:00:30:48:8b:84:35:08:00 SRC=55.X.Y.Z
DST=208.X.Y.Z LEN=401 TOS=0x00 PREC=0x00 TTL=55 ID=34299 PROTO=UDP
SPT=30757 DPT=1026 LEN=381

```

J'ai masqué son adresse IP car il s'agissait d'un paquet UDP, donc pas authentifié, même faiblement. Peut-être répondait-il tout simplement à un pirate qui usurpait mon adresse IP.

On peut ensuite regarder les statistiques, en demandant l'affichage de la chaîne `net2fw` (avec `Netfilter`, les règles de filtrage sont regroupées en chaînes). Au bout de quelques jours sur un petit serveur :

```

% sudo iptables -v -L net2fw
Chain net2fw (1 references)
pkts bytes target      prot opt in      out     source      destination      state RELATED,ESTABLISHED
 411K 163M ACCEPT      all  --  any     any     anywhere    anywhere
  12   720 ACCEPT      tcp  --  any     any     anywhere    anywhere          tcp dpt:6322
11477 588K ACCEPT      tcp  --  any     any     anywhere    anywhere          tcp dpt:smtp
 1500 89572 ACCEPT      tcp  --  any     any     anywhere    anywhere          tcp dpt:imap2
  296 17736 ACCEPT      tcp  --  any     any     anywhere    anywhere          tcp dpt:imaps
  381 19432 ACCEPT      tcp  --  any     any     anywhere    anywhere          tcp dpt:pop3
    2   112 ACCEPT      tcp  --  any     any     anywhere    anywhere          tcp dpt:pop3s
 1259 165K ACCEPT      icmp --  any     any     anywhere    anywhere          icmp echo-request
 1196 68996 ACCEPT      tcp  --  any     any     anywhere    anywhere          tcp dpt:www
  808 45296 ACCEPT      tcp  --  any     any     anywhere    anywhere          tcp dpt:https
 4206 287K ACCEPT      udp  --  any     any     anywhere    anywhere          udp dpt:domain
    11   664 ACCEPT      tcp  --  any     any     anywhere    anywhere          tcp dpt:domain
60921 7430K Drop        all  --  any     any     anywhere    anywhere
 1617 79082 LOG        all  --  any     any     anywhere    anywhere          LOG level info prefix

```

Notez le nombre de paquets jetés sans être signalés (car tellement fréquents qu'ils rempliraient les journaux).

Shorewall permet aussi de gérer facilement des listes noires. Ainsi, je peux mettre dans le fichier `blacklist` ces machines qui s'obstinent à utiliser mon serveur HTTP comme relais :

```
89.149.208.216/29
94.75.221.71
78.25.59.4
```

, je vérifie que mon fichier `interfaces` contient bien l'option `blacklist` :

```
net      eth0          detect      norfc1918,routefilter,dhcp,tcpflags,logmartians,nosmurfs,blacklist
```

je fais un `/etc/init.d/shorewall refresh` et ces trois machines (la première règle porte sur un réseau, donc plusieurs machines) sont désormais complètement interdites d'accès, ce que je peux vérifier avec `iptables` :

```
% sudo iptables -n -v -L blacklist
Chain blacklist (2 references)
 pkts bytes target      prot opt in      out     source           destination
  81  3888 DROP        all  --  *      *       89.149.208.216/29  0.0.0.0/0
  57  2736 DROP        all  --  *      *       94.75.221.71      0.0.0.0/0
 159  7632 DROP        all  --  *      *       78.25.59.4        0.0.0.0/0
```

Cela, c'était pour IPv4. En 2011, on veut aussi se prémunir contre d'éventuelles attaques sur IPv6. Shorewall permet également de générer des règles pour `ip6tables`. Cela se fait avec le paquetage `shorewall6` dont les fichiers de configuration sont quasiment les mêmes qu'avec IPv4 (par contre, il n'existe pas de mécanisme pour synchroniser les deux configurations, v4 et v6, c'est à vous de le faire à la main). Parmi les différences :

- Certaines options n'ont pas de sens en IPv6 comme `norfc1918`,
- ICMP doit s'écrire `ipv6-icmp` (ou bien le nom qui figure dans votre `/etc/protocols`, ou encore la valeur numérique, 58). Et les types n'ont pas les mêmes valeurs en v4 et v6 (par exemple, la demande d'écho, utilisée par ping, vaut 8 en v4 et 128 en IPv6).
- L'acceptation de l'ICMP est d'autant plus importante que Shorewall est livré avec des exemples incorrects, qui bloquent certains messages ICMP indispensables à la "*Path MTU discovery*" (type 2 / code 0, "*Packet Too Big*", voir le RFC 4443¹, section 3.2).

Une fois `shorewall6` lancé, vous pouvez voir les règles :

```
% sudo ip6tables -n -L net2fw|more
Chain net2fw (1 references)
target      prot opt source           destination
blacklst    all  ::/0             ::/0          ctstate INVALID,NEW
smurfs      all  ::/0             ::/0          ctstate INVALID,NEW
tcpflags    tcp  ::/0             ::/0
ACCEPT     all  ::/0             ::/0          ctstate RELATED,ESTABLISHED
ACCEPT     tcp  ::/0             ::/0          tcp dpt:9922
ACCEPT     tcp  ::/0             ::/0          tcp dpt:80
ACCEPT     tcp  ::/0             ::/0          tcp dpt:443
...
```

1. Pour voir le RFC de numéro NNN, <https://www.ietf.org/rfc/rfcNNN.txt>, par exemple <https://www.ietf.org/rfc/rfc4443.txt>

Est-ce qu'un pare-feu IPv6 sert à quelque chose aujourd'hui? L'examen des compteurs semble indiquer que non, pas pour l'instant. Après plusieurs heures :

```
% sudo ip6tables -v -n -L net2fw|more
...
1223 84896 ACCEPT      tcp      *      *      :::/0      :::/0      tcp dpt:80
...
2    632 DROP      all     *      *      :::/0      :::/0
```

Deux tentatives de passer outre, alors qu'IPv4 en a vu 3958 dans le même temps!

Avec ces règles, mon serveur est désormais raisonnablement protégé. Il y a bien sûr d'autres règles dans les applications elles-mêmes (par exemple, la directive AllowUsers dans la configuration d'OpenSSH). Rappelez-vous que la sécurité ne repose pas sur un seul truc mais sur une politique cohérente, avec défense en profondeur (c'est-à-dire qu'un maillon qui casse ne doit pas entraîner l'ouverture complète des portes).