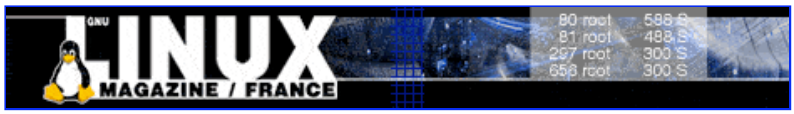

27 sept 2008

Filtrage des paquets avec le logiciel IPFilter

Catégorie : Sécurité Tags : misc

~~Retrouvez cet article dans : Misc 19~~

Le logiciel IPFilter – IPF en abrégé – est un logiciel permettant de doter un système Unix d'une fonction de Firewall. Contrairement au logiciel Netfilter qui opère dans le monde Linux, IPF est issu de la famille BSD. Il opère de ce fait sur FreeBSD, NetBSD mais aussi sur des Unix commerciaux comme IRIX, HP-UX, Solaris Sparc ou Solaris X86. Notons qu'il est présent par défaut dans la version 10 de Solaris pour remplacer le logiciel SunScreen.

Il est très simple de compiler ce firewall sous Solaris 8 et 9 et sa configuration ne présente pas de difficultés particulières. Il est néanmoins très fortement recommandé par Daren Reed, l'auteur de ce logiciel, de le compiler sur les architectures 64 bits avec le compilateur cc de Sun Studio plutôt que gcc.
La version 4 de ce logiciel est parue en 2004 et nous présenterons ici la version courante qui est la 4.1.8. L'ancienne version 3.4.35 continue son évolution de son côté.

Principales fonctions supportées par IPF

IPF est un logiciel de filtrage avec état (statefull) ce qui signifie qu'il mémorise des informations sur les connexions en maintenant une table d'état des connexions autorisées. Le maintien d'une table d'état permet d'autoriser automatiquement les paquets entrants correspondant à une connexion sortante autorisée et inversement. IPF n'est pas limité au filtrage de paquets et il assure aussi la translation d'adresses (NAT) et de ports.
IPF filtre les paquets au moyen de règles dans lesquelles on peut :

- faire référence aux adresses source et destination ;
- désigner l'interface à laquelle s'applique une règle ;
- indiquer le ou les ports et donc les protocoles concernés ;
- indiquer les options des paquets IP ;
- refuser des paquets fragmentés ou trop courts ;
- retourner ou non des indications via le protocole ICMP ;
- journaliser ;
- etc.

Nouveautés de la version 4

A partir de la version 4, IPF est composé de 2 paquetages qui s'installent et se mettent à jour de manière indépendante :

- PFil 2.1.6,
- IPFilter 4.1.8.

Le démarrage du filtre

Le module pfil est chargé dynamiquement au démarrage du système par le script /etc/rc2.d/S10pfil. Le filtrage est ensuite activé par le script /etc/rc2.d/S65ipfboot. Lors de l'installation, on pourra vérifier le chargement du module pfil à l'aide des deux commandes suivantes (le nom de l'interface réseau est ici hme0).

```
# strconf < /dev/hme
pfil
hme
# ifconfig hme0 modlist
0 arp
1 ip
2 pfil
3 hme
```

On voit avec cette deuxième commande que le module pfil s'insère entre la couche physique et la couche IP.

Le fichier de configuration iu.ap

Ce fichier est utilisé pour déclarer les interfaces pour lesquelles on va charger le module pfil. Dans le cas où une seule interface existe, on aura par exemple le contenu suivant :hme -1 0 pfil

Dans le cas de plusieurs interfaces, on aura autant de lignes que d'interfaces sur lesquelles on utilisera IPF.

La journalisation

IPF offre des possibilités de journalisation via le service syslog. Il permet de choisir le niveau de journalisation pour chaque règle. Dans les exemples de cet article, on utilise le niveau local0.notice. La journalisation sera donc effective si on ajoute dans le fichier de configuration /etc/syslog.conf la ligne suivante et si on signale le changement au démon syslogd.

```
local0.notice /var/log/ipfilog
```

Dans les exemples qui suivent, on journalise les échecs et on utilise un seul niveau de journalisation. En pratique, il faudra paramétrer la journalisation en fonction de l'utilisation et du trafic pour faire en sorte que les données générées restent exploitables.

Le fichier de configuration principal ipf.conf

Le fichier ipf.conf décrit les règles de filtrage. Les règles sont parcourues en séquence de la première à la dernière. Imaginons que ce fichier ne comporte que les 2 règles suivantes :

```
block in all
pass in all
```

Lorsqu'un paquet IP entrant arrive, il est reconnu par la première règle qui le bloque. Puis on passe à la règle suivante qui le laisse passer. C'est la dernière règle applicable à un paquet qui est retenue et donc on laisse passer les paquets entrants. Lorsqu'un paquet est reconnu par une règle comportant la directive quick, les règles qui suivent sont ignorées. L'emploi de quick détermine l'ordonnancement des règles. Ici la directive block peut être placée avant la directive pass car elle n'utilise pas la directive quick :

```
block in on hme0 all
pass in quick on hme0 proto tcp from any to 172.24.34.1/32 port = 80
...
```

Cas d'un poste de travail seulement client

Nous allons introduire la syntaxe utilisée par IPF par un exemple simple consistant à protéger un poste de travail du monde extérieur grâce à ce logiciel. Le poste de travail doit interdire l'accès à tous ses serveurs et autoriser l'utilisation en mode client sans restriction. Pour une protection déjà efficace les quelques lignes suivantes suffisent :

```
block in log level local0.notice quick on hme0 all
pass out quick on hme0 proto tcp from any to any keep state
pass out quick on hme0 proto udp from any to any keep state
pass out quick on hme0 proto icmp from any to any keep state
block out log level local0.notice on hme0 all
```

Ces 5 règles n'autorisent aucun accès entrant mais permettent d'utiliser la plupart des logiciels client. Examinons-les plus en détail. La première ligne interdit tout paquet entrant sur l'interface hme0 et comme elle comporte la directive ~~quick~~, on ne va pas plus loin dans le parcours des règles du moins pour les paquets entrants. Toutes les tentatives de connexion à un service du poste de travail seront donc interdites et de surcroît enregistrées par syslog au niveau ~~local0.notic~~e. La deuxième ligne autorise les paquets TCP sortant de l'interface hme0 avec n'importe quelle adresse source vers n'importe quelle adresse destination et utilisant des ports source et destination quelconques. La directive ~~keep state~~ est très importante, car c'est elle qui indique qu'on maintient une table d'état pour les connexions. Une fois la connexion établie, c'est cette directive qui autorise les paquets entrants correspondant à une connexion sortante. Sans elle, il serait nécessaire de rajouter une règle pour autoriser les paquets entrants associés à cette connexion sortante. La ligne trois et la ligne quatre jouent le même rôle respectivement pour les protocoles UDP et ICMP. La dernière ligne bloque tout paquet sortant non autorisé par les 3 lignes qui précèdent.
Pour faire prendre en compte ces règles, le plus simple dans un premier temps consiste à arrêter puis à relancer le filtre soit :

```
/etc/rc2..d/SS6ipfboot stop
/etc/rc2..d/SS6ipfboot start
```

On aurait pu être plus précis dans l'écriture de ces règles dans le cas où l'adresse associée à l'interface est connue (ce n'est pas toujours le cas, par exemple dans le cas d'une connexion PPP ou celui d'une adresse obtenue dynamiquement par une requête DHCP). Soit 172.24.34.1 l'adresse de l'interface hme0. Les règles précédentes peuvent alors se réécrire de la façon suivante :

```
block in log level local0.notice quick on hme0 all
pass out quick on hme0 proto tcp from 172.24.34.1/32 to any keep state
pass out quick on hme0 proto udp from 172.24.34.1/32 to any keep state
pass out quick on hme0 proto icmp from 172.24.34.1/32 to any keep state
block out log level local0.notice on hme0 all
```

On sait que la grande majorité des applications TCP utilisent un port source supérieur à 1024. On pourrait donc encore modifier les règles en définissant une restriction sur les ports TCP.

```
block in log level local0.notice quick on hme0 all
pass out quick on hme0 proto tcp from 172.24.34.1/32 port > 1024 to any keep state
pass out quick on hme0 proto udp from 172.24.34.1/32 to any keep state
pass out quick on hme0 proto icmp from 172.24.34.1/32 to any keep state
block out log level local0.notice on hme0 all
```

Il y a toutefois des exceptions avec les r-commandes qui utilisent des ports source décroissants à partir de 1023. Il faudra donc rajouter des règles spécifiques dans le cas où on désire utiliser les logiciels correspondants. Notons qu'on pourrait être encore plus précis en indiquant les adresses et les ports de sorties autorisés. En pratique, il vaut mieux rester assez vague si on souhaite que la gestion des règles garde une certaine simplicité et pour ne pas les alourdir d'une multitude de cas particuliers.
Pour autoriser ~~rlogin~~ et ~~rsh~~ clients, l'ajout des 2 lignes suivantes serait nécessaire :

```
pass out quick on hme0 proto tcp from 172.24.34.1/32 port < 1024 to any port = 513 keep state
pass out quick on hme0 proto tcp from 172.24.34.1/32 port < 1024 to any port = 514 keep state
```

Ou plus simplement :

```
pass out quick on hme0 proto tcp from 172.24.34.1/32 port < 1024 to any port 513 >< 515 keep state
```

Protocoles particuliers

Pour définir les règles, une bonne connaissance des protocoles qu'on veut filtrer est nécessaire. Le protocole FTP par exemple est très particulier puisqu'il utilise à la fois le port 21 (FTP) pour la connexion et le port 20 (FTP-data) dès qu'on utilise une commande qui effectue un transfert de données (dir, get, put, etc.). Lorsqu'on exécute une telle commande, il y a dans un premier temps une négociation de port entre le client et le serveur puis le serveur transfère depuis le port 20 vers un port supérieur à 1024 du client obtenu par la négociation précédente. Autrement dit, une fois la négociation de port terminée le client se comporte pour le transfert de données comme un serveur. De ce fait pour le bon fonctionnement d'un client FTP en « mode actif », l'ajout de la ligne suivante est obligatoire :

```
pass in quick on hme0 proto tcp from any port = 20 to 172.24.34.1/32 port > 1024 keep state
```

Notons que la plupart des clients récents (les navigateurs en particulier) supportent un autre mode de connexion appelé « mode passif ». En mode passif, la valeur 20 n'est plus utilisée et le port FTP-data est un port de valeur supérieure à 1024 obtenu par négociation. En outre, c'est le client qui prend l'initiative de la connexion vers ce port FTP-data. En mode passif, la règle précédente n'est plus nécessaire.
On retiendra de ce cas particulier que le protocole FTP est un protocole particulier et que si le mode passif simplifie le filtrage côté client, il le complique du côté serveur.

Cas d'un poste de travail client et serveur

Imaginons maintenant que le poste client devienne aussi serveur et souhaite ouvrir les accès ssh pour l'ensemble des postes de son réseau local et autorise la réception de messages SMTP depuis la passerelle de messagerie dont l'adresse est 172.24.1.25.
L'ouverture de ces 2 services se traduit par l'ajout de 2 règles qui seront placées avant la règle qui bloque les paquets entrants. Ces règles utilisent obligatoirement la directive ~~quick~~ pour interrompre le traitement et la directive ~~keep state~~ pour autoriser les paquets en retour correspondants. De plus comme une ouverture de connexion TCP positionne le flag SYN, on n'autorise la connexion que si ce flag est positionné.
On remarquera que le protocole applicatif est désigné soit par son numéro de port officiel soit par le nom qui lui est associé dans le fichier ~~/etc/services~~.

```
pass in quick on hme0 proto tcp from 172.24.0.0/16 to any port = 22 flags S keep state
pass in quick on hme0 proto tcp from 172.24.1.25/32 to any port = smtp flags S keep state
block in log level local0.notice quick on hme0 all
pass out quick on hme0 proto tcp from any port to any keep state
pass out quick on hme0 proto udp from any port to any keep state
pass out quick on hme0 proto icmp from any port to any keep state
block out log level local0.notice on hme0 all
```

Avec un filtre ordinaire sans état, la première règle devrait être remplacée par les 2 règles suivantes :

```
pass in quick on hme0 proto tcp from 172.24.0.0/16 to any port = 22 flags S
pass out quick on hme0 proto tcp from any port = 22 to 172.24.0.0/16 port > 1024
```

Outre le fait qu'il faut écrire une deuxième règle pour le paquet sortant, cette règle autorisant le paquet entrant correspondant est moins précise pour le numéro de port que celle automatiquement mise en place quand on utilise le mode avec état (~~keep state~~). Avec état, seul le port effectivement utilisé est autorisé, sans état ce sont tous les ports supérieurs à 1024. En outre, la sortie de paquets depuis le port 22 est possible que la connexion ssh existe ou non.

Fonctionnalités avancées

L'anti-spoofing

Il n'est pas normal que des paquets avec des adresses privées telles que celles définies par le RFC1918 se présentent en entrée sur une interface reliée au monde extérieur. Il est donc souhaitable dans un réseau utilisant un adressage privé d'interdire sur l'interface Internet l'entrée de tels paquets.

```
block in quick on sppp0 from 192.168.0.0/16 to any
block in quick on sppp0 from 172.16.0.0/16 to any
block in quick on sppp0 from 10.0.0.0/8 to any
block in quick on sppp0 from 127.0.0.0/8 to any
block in quick on sppp0 from 0.0.0.0/8 to any
```

Filtrage de paquets particuliers

On pourra filtrer les paquets qui présentent certaines particularités parfois détournées de leur fonction initiale pour contourner les règles du coupe-feu. On pourra ainsi rejeter tout paquet fragmenté :

```
block in all with frag
```

ou ne rejeter que les paquets TCP fragmentés trop courts :

```
block in log quick proto tcp all with short
```

rejeter tout paquet contenant des options IP :

```
block in log all with ipopts
```

ou indiquer les options qui seront filtrées :

```
block in quick all with opt lsrr
block in quick all with opt ssrr
```

Répondre ou ne pas répondre

Lorsqu'on bloque un paquet, on peut indiquer clairement le refus du paquet par un reset de connexion ou retourner des informations via ICMP. La politique à adopter sera déterminée par la visibilité qu'on voudra donner de l'équipement filtrant au monde extérieur. On pourra par exemple offrir une bonne visibilité du monde Intranet et être totalement obscur du côté Internet. IPF, comme le montrent ces quelques exemples, offre une grande souplesse dans la définition d'une telle politique.
Refuser toute requête ICMP sauf celles qui entrent sur l'interface hme0.
block in proto icmp all
pass in on hme0 proto icmp from any to any icmp-type echo keep state
Retourner un refus de connexion immédiat pour les paquets qui arrivent sur une interface et ne donner aucune indication pour une autre.

```
block return-rst in log local0.notice on hme0 proto tcp from 192.168.0.0/24 to any port = 22
block in log quick on hme1
```

Retourner des indications via ICMP.

```
block return-icmp-as-dest(port-unr) in log on hme0 proto udp from any to any port = 111
```

La translation d'adresses et la translation de ports

Les fonctionnalités d'IPF ne sont pas limitées au filtrage des paquets. La translation d'adresses et/ou la translation de ports sont des opérations qui doivent être déclarées dans le fichier ~~ipnat.conf~~. On dispose pour cela de 3 directives :

- ~~rdr~~ translate l'adresse et/ou le port de paquets entrants,
- ~~map~~ translate l'adresse et/ou le port de paquets sortants,
- ~~bitmap~~ translate les paquets qui entrent et qui sortent d'une interface.

Translation des adresses sortantes

Le fichier ~~ipnat.conf~~ sera utilisé pour définir les règles de translation d'adresses et/ou des règles de translation de ports. Au niveau de la translation d'adresses, on peut translater :

- ~~n~~ adresses en une seule adresse,
- ~~n~~ adresses vers p adresses,
- ~~n~~ adresses vers n adresses.

Les équipements client internes qui voudront bénéficier de la translation d'adresses devront déclarer l'équipement IPF comme routeur par défaut dans le cas où la translation est généralisée ou déclarer une route qui transite par le routeur

IPF vers l'équipement externe ciblé. En outre, il est aussi indispensable de faire pointer le client DNS vers un serveur DNS sachant résoudre les adresses internes privées comme les adresses externes dans le cas où on généralise la translation ou déclarer l'adresse ciblée dans le fichier ~~/etc/hosts~~.
On devra par ailleurs s'assurer que l'équipement IPF route bien les paquets. Si ce n'est pas le cas et dans le cas de Solaris ces deux commandes sont à exécuter :

```
ndd -set /dev/ip ipforwarding 1
ndd -set /dev/ip ip_strict_dst_multihoming 1
```

Translation n vers une

On translate ici n adresses vers une seule adresse, généralement l'adresse de l'interface Internet, comme le fait un mandataire web au niveau applicatif, mais avec beaucoup plus d'efficacité puisqu'on agit ici entre le niveau physique et la couche IP.

```
map hme1 192.168.1.0/24 -> 192.18.110.1/32
```

Dans le cas où on ne connaîtrait pas à l'avance l'adresse de l'interface de sortie (connexion PPP par exemple) ou s'il y a une seule possibilité d'interface de sortie (l'équipement sur lequel opère IPF ne dispose que de 2 interfaces), on écrira cette règle ainsi :

```
map hme1 192.168.1.0/24 -> 0/32
```

On pourra aussi translater les ports source en fixant une plage de ports à utiliser (la deuxième ligne concerne les protocoles autres que TCP et UDP comme ICMP ou ESP/AH par exemple):

```
map hme1 192.168.1.0/24 -> 0/32 portmap tcp/udp 25000:35000
map hme1 192.168.1.0/24 -> 0/32
```

Translation n vers p

```
map hme1 192.168.0.0/16 -> 192.18.110.0/24
```

Translation n vers n

```
map hme1 192.168.1.0/24 -> 192.18.110.0/24
```

Ces règles ne sont pas suffisantes pour que la translation d'adresses fonctionne et il est nécessaire comme on l'a déjà indiqué :

- que le coupe-feu IPF soit désigné par les postes de travail désirant utiliser le NAT comme le routeur ;
- que l'IP forwarding soit établi au niveau du coupe-feu.

Mais ce n'est pas encore suffisant et il faut aussi autoriser la sortie des paquets au niveau du filtre IP et donc rajouter dans ~~ipf.conf~~ les règles suivantes :

```
pass out on hme1 quick proto tcp/udp from 192.168.1.0/24 to any keep state
pass out on hme1 quick proto any from 192.168.1.0/24 to any keep state
```

Ces règles concernant l'interface de sortie côté Internet peuvent sembler curieuses. C'est bien l'adresse d'origine qu'il faut filtrer et cela signifie tout simplement que la translation est faite après le filtrage.
Dans le cas où l'adresse translatée est différente de l'adresse de l'interface de sortie, il y a une dernière condition à remplir pour que le NAT fonctionne. Il faut faire du proxy ARP pour cette adresse en lui associant l'adresse physique de l'interface de sortie. Si par exemple l'adresse de l'interface de sortie est 192.18.110.1 et que son adresse physique est 00:03:bb:18:96:15 et si l'adresse NAT utilisée est 192.18.110.2, il faudra publier l'adresse avec la commande ~~arp~~ soit :

```
arp -s 192.18.110.2 00:03:bb:18:96:15 pub
```

Bien évidemment si la translation concerne n adresses, on devra répéter cette association pour les n adresses utilisées. Si elles sont continues, on pourra faire cela avec simplement un script bash.

```
#!/bin/bash
```

```
for (( i =2; i <= 255; i++ ))
do
arp -s 192.18.110.$i 00:03:bb:18:96:15 pub
done
```

Il est possible de conditionner la translation d'adresses en fonction de l'adresse source et de l'adresse destination :

```
map hme1 from 192.168.1.10/32 to 223.223.10.0/24 -> 192.18.110.1/32
```

Translation des adresses et ports entrants

On peut effectuer une translation de port pour un serveur HTTP interne et le rendre visible de l'extérieur. Ainsi le serveur web interne s'exécutant sur le port 8080 apparaîtra au monde externe avec l'adresse 192.168.110.1 et le port 80.

```
rdr hme1 192.168.110.1/32 port 80 -> 192.168.1.27 port 8080
```

Dans le cas de la translation d'adresses et de ports entrants, la translation a lieu avant le filtrage et il faudra donc en tenir compte pour filtrer les accès internes. Ainsi, si on souhaite rendre ce serveur accessible de l'extérieur par un client d'adresse IP 223.223.10.1, on utilisera la règle de filtrage suivante :

```
pass in on hme1 proto tcp from 223.223.10.1/32 to 192.168.1.27/32 port = 8080 \ flags S keep state
```

Il est enfin possible de conditionner la translation en fonction de l'adresse source :

```
rdr hme1 from 223.223.10.2/32 to 192.168.110.1/32 port 80 -> 192.168.1.27 port 8080
```

Proxy transparent

Lorsqu'on translate les adresses sortantes des clients externes, HTTP ou FTP joignent directement Internet avec une adresse apparente définie par le coupe-feu. Il n'est donc plus nécessaire de déclarer un proxy au niveau de ces clients HTTP et FTP. Toutefois, dans le cas particulier de ces deux protocoles, ce mode de fonctionnement constitue une régression en termes de sécurité par rapport à un mandataire web. En effet, les mandataires HTTP et FTP assurent généralement un filtrage très poussé allant du filtrage d'URL, au rejet de certains types de fichiers en passant par la décontamination virale. Il est donc souhaitable de continuer à faire passer ces protocoles par un mandataire et il est souhaitable de l'effectuer en laissant les clients dans la configuration par défaut qui n'utilise pas un mandataire.
Dans le cas d'un mandataire FTP, tout paquet arrivant sur l'interface Intranet vers n'importe quelle adresse port 21 sera redirigé sur l'adresse du mandataire local opérant sur le port 21 (notons à ce sujet qu'il n'est pas possible de rediriger sur l'adresse 127.0.0.1 car l'interface loopback Solaris n'est pas, pour des raisons de performance, implantée de façon standard).

```
rdr hme0 0.0.0.0/0 port 21 -> 192.168.1.10 port 21
```

La définition de groupes d'adresses ippool.conf

Ce fichier constitue l'une des nouveautés de la version 4. Il permet de définir un ensemble d'adresses et de faire référence dans le fichier ~~ipf.conf~~ à des adresses définies dans ~~ippool.conf~~ avec la syntaxe ~~pool/n~~ ou ~~n~~ est le numéro du groupe.

```
#
# /etc/opt/ipf/ippool.conf
#

table role = ipf type = tree number = 10
{
10.1.1.11/32;
10.1.1.23/32;
10.20.0.0/16;
!10.20.40.0/24
};

table role = ipf type = tree number = 11
{
172.24.0.0/16;
```

```
10.1.1.219/32;
10.1.15.0/24;
};
```

On fera référence à un ensemble d'adresses ainsi définies de la façon suivante :

```
pass in quick on hme0 proto tcp from pool/10 to any port = 22 flags S keep state
```

Ce qui présente l'avantage de remplacer plusieurs lignes par une seule. Il permet donc de rendre le fichier ipf.conf beaucoup plus lisible et on n'hésitera pas à utiliser cette possibilité.

Mise au point et audit du filtrage

Définir les règles de filtrage est une opération relativement aisée lorsqu'on protège un poste de travail doté d'une seule interface IP. Il est probablement utile de commencer l'apprentissage de ce logiciel en commençant par ce cas simple, puis de passer à celui d'un serveur doté d'une seule interface avant de s'attaquer au cas plus général d'un équipement multidomicilié.

Lorsque le filtrage mis en place ne fonctionne pas avec certains protocoles, les causes du dysfonctionnement se trouvent dans les informations journalisées si toutefois on a bien pris la peine de journaliser les échecs. L'examen du journal avant de se lancer dans la modification de règles est donc indispensable. Si on connaît mal le protocole concerné, il ne sera pas inutile de relire son RFC.

Pour un serveur équipé de plusieurs interfaces jouant le rôle de coupe-feu, le risque d'erreur croît avec le nombre d'interfaces, le nombre de protocoles et le nombre d'adresses autorisées. La journalisation est une aide précieuse pour la mise au point. Elle enregistre aussi toute tentative illégale. Si le rejet des tentatives illégales est une indication de bon fonctionnement du filtre, il ne garantit en rien que celui-ci soit conforme au cahier des charges.

Il en résulte qu'un contrôle de l'efficacité des règles est indispensable. Pour être valable, ce test devra être répété autant de fois que la machine possède d'interfaces IP en se plaçant dans les conditions adéquates. Le logiciel nmap est tout à fait indiqué pour valider le filtrage et garantir qu'il soit conforme à la politique de sécurité du site.

Conclusion

Cette présentation décrit de manière succincte les fonctions principales du pare-feu IPF. L'aspect exploitation de ce logiciel a été presque passé sous silence et on aurait pu indiquer comment ajouter des règles à la volée ou examiner les statistiques. C'est l'aspect configuration qui a été privilégié. On pourra l'approfondir en consultant les pages du manuel qui décrivent à l'aide d'une grammaire BNF les règles utilisables dans les différents fichiers de configuration. On y découvrira comment IPF est capable de retourner n'importe quel code ICMP, de sélectionner les paquets en fonction des options IP, des niveaux de sécurité, du TTL ou encore des flags TCP.

Bibliographie

- IP Filter Based Firewalls HOWTO : Brendan Conoboy et Erik Fichtner
- IP Filter FAQ : Phil Dibowitz
- Solaris IP Filter Overview
- Pages du manuel ~~ipf(5)~~, ~~ippool.conf(5)~~, ~~ipnat.conf(5)~~

~~Retrouvez cet article dans : **Misc 19**~~

Posté par (La rédaction) | Signature : Christian Pélissier | Article paru dans 

Laissez une réponse

Vous devez avoir ouvert une session pour écrire un commentaire.

« Précédent Aller au contenu »

Articles de 1ère page

- réception d'images satellites : utilisation d'un système embarqué
- réception d'images satellites : principes de base
- Les chantiers OpenBSD
- Pour quelques bits d'information
- Linux embarqué : BusyBox « in a nutshell »
- La boîte à outils libres pour l'embarqué
- Sécurité avancée du serveur web Apache : mod_security et mod_dosevasive
- Quelques éléments de sécurité des réseaux privés virtuels MPLS/VPN
- Quelles solutions pour Linux embarqué ?
- Les systèmes embarqués : une introduction

Il y a actuellement

- 811 articles/billets en ligne.

Catégories

- Administration réseau
- Administration système
- Agenda-Interview
- Audio-vidéo
- Bureautique
- Comprendre

- Distribution
- Embarqué
- Environnement de bureau
- Graphisme
- Jeux
- Matériel
- News
- Programmation
- Réfléchir
- Sécurité
- Utilitaires
- Web

Archives

- octobre 2008
- septembre 2008
- août 2008
- juillet 2008
- juin 2008
- mai 2008
- avril 2008
- mars 2008
- février 2008
- janvier 2008
- décembre 2007
- novembre 2007
- février 2007

GNU/Linux Magazine

- EuroBSDCon 2008 à Strasbourg 18 et 19 octobre
- GNU/Linux Magazine N°109 - Octobre 2008 - Chez votre marchand de journaux
- Édito : GNU/Linux Magazine 109
- GLMF, partenaire de l'évènement "Paris, capitale du Libre"
- GNU/Linux Magazine 108 - Septembre 2008 - Chez votre marchand de journaux

GNU/Linux Pratique

- EuroBSDCon 2008 à Strasbourg 18 et 19 octobre
- Linux Pratique Essentiel N°4 - Octobre/Novembre 2008 - Chez votre marchand de journaux
- Édito : Linux Pratique Essentiel N° 4
- Linux Pratique Essentiel N°4 : Références des articles
- Linux Pratique Essentiel 4 - Communiqué de presse

MISC Magazine

- Misc 39 : Fuzzing - Injectez des données et trouvez les failles cachées - Septembre/Octobre 2008 - Chez votre marchand de journaux
- Edito : Misc 39
- MISC 39 - Communiqué de presse
- Salon Infosecurity & Storage expo - 19 et 20 novembre 2008.
- Misc 38 : Codes Malicieux, quoi de neuf ? - Juillet/Août 2008 - Chez votre marchand de journaux