PF.CONF(5) Manuel du programmeur OpenBSD

NOM

pf.conf - fichier de configuration du filtre de paquets Packet Filter.

DESCRIPTION

Le filtre de paquets pf(4) modifie, bloque ou laisse passer les paquets selon des règles ou des définitions spécifiées dans **pf.conf**.

ORDRE DES DÉCLARATIONS

Il y a sept types de déclaration dans **pf.conf** :

- **Macros**

Des variables peuvent être définies par l'utilisateur et utilisées plus loin, ce qui simplifie le fichier de configuration. Les macros doivent être définies avant d'être référencées dans **pf.conf**.

- **Tables**

Les tables fournissent un mécanisme d'amélioration des performances et de la flexibilité des règles possédant de grands nombres d'adresses source ou destination.

- **Options**

Les options affinent le comportement du moteur de filtrage de paquets.

- **Normalisation du trafic** (c'est-à-dire *scrub*)

La normalisation du trafic protège les machines internes contre les incohérences dans les protocoles d'Internet, et leurs implémentations.

- **File d'attente**

La mise en file d'attente fournit un contrôle de bande-passante basé sur des règles.

- **Traduction** (diverses formes de NAT)

Les règles de traduction spécifient les correspondances et les redirections entre adresses.

- **Filtrage de paquet**

Le filtrage de paquets consiste à décider si un paquet est bloqué ou pas, en se basant sur des règles.

À l'exception des **macros** et des **tables**, les types de déclarations devraient être groupés et apparaître dans **pf.conf** dans l'ordre montré ci-dessus, car cela correspond à l'ordre du traitement

sous-jacent opéré par le moteur de filtrage de paquet. Par défaut, pfctl(8) s'assure de cet ordre (voir *set require-order* ci-dessous).

Les commentaires peuvent être placés n'importe où dans le fichier en utilisant un dièse (“#”), et ils s'étendent jusqu'à la fin de la ligne courante.

Des fichiers de configuration supplémentaires peuvent être inclus au moyen du mot-clef **include**, par exemple :

```
        include "/etc/pf/sub.filter.conf"
```

MACROS

On peut définir des macros qui seront par la suite développées dans le contexte. Les noms de macros doivent commencer par une lettre, et peuvent contenir des lettres, des chiffres et des soulignés. Les noms de macros ne peuvent être des mots réservés (par exemple *pass*, *in*, *out*). Les macros ne sont pas développées à l'intérieur de guillemets.

Par exemple,

```
        ext_if = "kue0"
        all_ifs = "{" $ext_if lo0 "}"
        pass out on $ext_if from any to any
        pass in  on $ext_if proto tcp from any to any port 25
```

TABLES

Les tables sont des structures nommées pouvant contenir un ensemble d'adresses et de réseaux. Les recherches dans les tables pf(4) sont relativement rapides, rendant une unique règle utilisant des tables bien plus efficace, en terme d'utilisatoin du processeur et de consommation de mémoire, qu'un grand nombre de règles qui ne diffèrent que par l'adresse IP (qu'elle soit créée explicitement ou automatiquement par développement de règle).

Les tables peuvent être utilisées comme source ou destination de règles de filtrage, de règles *scrub* ou de règles de traduction comme *nat* ou *rdr* (voir ci-dessous pour des détails sur les divers types de règles). Les tables peuvent également être utilisées en tant qu'adresse de redirection des règles *nat* et *rdr*, et dans les options de routage des règles de filtrage, mais uniquement pour les pools *round-robin*.

OPTIONS

NORMALISATION DU TRAFIC

FILE D'ATTENTE

TRADUCTION

FILTRAGE DE PAQUET

PARAMÈTRES

ROUTAGE

OPTIONS DE LA RÉSERVE

MODULATION D'ÉTAT

PROXY SYN

TRAÇAGE AVEC SUIVI D'ÉTAT

SYSTÈME D'EXPLOITATION

BLOQUER LES USURPATIONS

GESTION DES FRAGMENTS

ANCRES

EXEMPLES DE TRADUCTION

EXEMPLES DE FILTRES

```
# L'interface externe est kue0
# (157.161.48.183, la seule adresse routable)
```

```
   # et le réseau privé est 10.0.0.0/8, pour lequel nous faisons de la NAT.

   # utilise une macro pour le nom de l'interface, de manière à ce qu'il
puisse être changé facilement
   ext_if = "kue0"

   # normalise tout le trafic entrant
   scrub in on $ext_if all fragment reassemble

   # bloque et journalise tout par défaut
   block return log on $ext_if all

   # bloque tout ce qui vient de sources pour lesquelles nous n'avons pas de
route de retour
   block in from no-route to any

   # bloque les paquets dont l'interface ingress ne correspond pas à celle
de
   # la route de retour vers l'adresse source
   block in from urpf-failed to any

   # bloque et journalise les paquets sortants qui n'ont pas notre adresse
pour source,
   # soit ils sont usurpés soit il y a une mauvaise configuration (NAT
désactivée,
   # par exemple), nous voulons être sympa et ne pas expédier des déchets.
   block out log quick on $ext_if from ! 157.161.48.183 to any

   # laisse silencieusement tomber les broadcasts (bruit de modem câble)
   block in quick on $ext_if from any to 255.255.255.255

   # bloque et journalise les paquets entrants provenant d'adresses
réservées et invalides,
   # ils sont soit usurpés soit mal configurés, nous ne pouvons de toute
façon pas y répondre
   # (d'où pas de rst en retour).
   block in log quick on $ext_if from { 10.0.0.0/8, 172.16.0.0/12, \
         192.168.0.0/16, 255.255.255.255/32 } to any

   # ICMP

   # laisse passer en entrée/sortie certaines requêtes ICMP et conserve
l'état (ping)
   # La correspondance de l'état est basée sur l'adresse des hôtes et les ID
ICMP (pas sur le type/code),
   # donc les réponses (comme 0/0 pour 8/0) correspondra aux requêtes
   # Les messages d'erreur ICMP (qui se réfère toujours à un paquet TCP/UDP)
sont
   # gérés par les états TCP/UDP
```

```
   pass on $ext_if inet proto icmp all icmp-type 8 code 0

   # UDP

   # laisse passer en sortie toutes les connexions UDP et conserve l'état
   pass out on $ext_if proto udp all

   # laisse passer en entrée certaines connexions UDP et conserve l'état
(DNS)
   pass in on $ext_if proto udp from any to any port domain

   # TCP

   # laisse passer en sortie toutes les connexions TCP et module l'état
   pass out on $ext_if proto tcp all modulate state

   # laisse passer en entrée certaines connexions TCP et conserve l'état
(SSH, SMTP, DNS, IDENT)
   pass in on $ext_if proto tcp from any to any port { ssh, smtp, domain, \
         auth }

   # N'autorise pas les connexions SMTP Windows 9x car il s'agit typiquement
de
   # vers viraux. Nous pourrions également procéder en limitant ces OS à une
connexion chacun.
   block in on $ext_if proto tcp from any os {"Windows 95", "Windows 98"} \
         to any port smtp

   # IPv6
   # laisse passer en entrée/sortie tout le trafic : remarquez que nous
devons activer ceci,
   # deux fois, une pour notre interface physique et une autre pour notre
tunnel
   pass quick on gif0 inet6
   pass quick on $ext_if proto ipv6

   # Étiquetage de paquet

   # trois interfaces : $int_if, $ext_if et $wifi_if (sans fil). La NAT
   # est effectuée sur $ext_if pour tous les paquets sortants. Étiquette les
paquets
   # en entrée sur $int_if et laisse passer en sortie ces paquets étiquetés
sur $ext_if.
   # Tous les autres paquets sortants (c'est-à-dire les paquets provenant du
réseau sans fil)
   # ne sont autorisés à accéder qu'au port 80.

   pass in on $int_if from any to any tag INTNET
   pass in on $wifi_if from any to any
```

```
   block out on $ext_if from any to any
   pass out quick on $ext_if tagged INTNET
   pass out on $ext_if proto tcp from any to any port 80
```

```
   # Étiquette les paquets entrants qui sont redirigés vers
[[:maxime:openbsd:manpages-fr:8:spamd|spamd(8)]].
   # Utilise l'étiquette pour laisser passer ces paquets au travers du
filtre de paquets.
```

```
   rdr on $ext_if inet proto tcp from <spammers> to port smtp \
           tag SPAMD -> 127.0.0.1 port spamd
```

```
   block in on $ext_if
   pass in on $ext_if inet proto tcp tagged SPAMD
```

GRAMMAIRE

La syntaxe de **pf.conf** en forme BNF :

```
   line           = ( option | pf-rule | nat-rule | binat-rule | rdr-rule |
                    antispoof-rule | altq-rule | queue-rule | trans-anchors
|
                    anchor-rule | anchor-close | load-anchor | table-rule |
                    include )
```

```
   option         = "set" ( [ "timeout" ( timeout | "{" timeout-list "}" ) ]
|
                    [ "ruleset-optimization" [ "none" | "basic" | "profile"
]] |
                    [ "optimization" [ "default" | "normal" |
                    "high-latency" | "satellite" |
                    "aggressive" | "conservative" ] ]
                    [ "limit" ( limit-item | "{" limit-list "}" ) ] |
                    [ "loginterface" ( interface-name | "none" ) ] |
                    [ "block-policy" ( "drop" | "return" ) ] |
                    [ "state-policy" ( "if-bound" | "floating" ) ]
                    [ "state-defaults" state-opts ]
                    [ "require-order" ( "yes" | "no" ) ]
                    [ "fingerprints" filename ] |
                    [ "skip on" ifspec ] |
                    [ "debug" ( "none" | "urgent" | "misc" | "loud" ) ] )
```

```
   pf-rule        = action [ ( "in" | "out" ) ]
                    [ "log" [ "(" logopts ")"] ] [ "quick" ]
                    [ "on" ifspec ] [ "fastroute" | route ] [ af ] [
protospec ]
                    hosts [ filteropt-list ]
```

```
   logopts        = logopt [ "," logopts ]
   logopt         = "all" | "user" | "to" interface-name
```

```
   filteropt-list = filteropt-list filteropt | filteropt
   filteropt      = user | group | flags | icmp-type | icmp6-type | "tos"
tos |
                    ( "no" | "keep" | "modulate" | "synproxy" ) "state"
                    [ "(" state-opts ")" ] |
                    "fragment" | "no-df" | "min-ttl" number | "set-tos" tos
|
                    "max-mss" number | "random-id" | "reassemble tcp" |
                    fragmentation | "allow-opts" |
                    "label" string | "tag" string | [ ! ] "tagged" string |
                    "queue" ( string | "(" string [ [ "," ] string ] ")" ) |
                    "rtable" number | "probability" number"%"
```

```
   nat-rule       = [ "no" ] "nat" [ "pass" [ "log" [ "(" logopts ")" ] ] ]
                    [ "on" ifspec ] [ af ]
                    [ protospec ] hosts [ "tag" string ] [ "tagged" string ]
                    [ "->" ( redirhost | "{" redirhost-list "}" )
                    [ portspec ] [ pooltype ] [ "static-port" ] ]
```

```
   binat-rule     = [ "no" ] "binat" [ "pass" [ "log" [ "(" logopts ")" ] ]
]
                    [ "on" interface-name ] [ af ]
                    [ "proto" ( proto-name | proto-number ) ]
                    "from" address [ "/" mask-bits ] "to" ipspec
                    [ "tag" string ] [ "tagged" string ]
                    [ "->" address [ "/" mask-bits ] ]
```

```
   rdr-rule       = [ "no" ] "rdr" [ "pass" [ "log" [ "(" logopts ")" ] ] ]
                    [ "on" ifspec ] [ af ]
                    [ protospec ] hosts [ "tag" string ] [ "tagged" string ]
                    [ "->" ( redirhost | "{" redirhost-list "}" )
                    [ portspec ] [ pooltype ] ]
```

```
   antispoof-rule = "antispoof" [ "log" ] [ "quick" ]
                    "for" ifspec [ af ] [ "label" string ]
```

```
   table-rule     = "table" "<" string ">" [ tableopts-list ]
   tableopts-list = tableopts-list tableopts | tableopts
   tableopts      = "persist" | "const" | "counters" | "file" string |
                    "{" [ tableaddr-list ] "}"
   tableaddr-list = tableaddr-list [ "," ] tableaddr-spec | tableaddr-spec
   tableaddr-spec = [ "!" ] tableaddr [ "/" mask-bits ]
   tableaddr      = hostname | ifspec | "self" |
                    ipv4-dotted-quad | ipv6-coloned-hex
```

```
   altq-rule      = "altq on" interface-name queueopts-list
                    "queue" subqueue
```

```
   queue-rule     = "queue" string [ "on" interface-name ] queueopts-list
                    subqueue
```

```
   anchor-rule    = "anchor" [ string ] [ ( "in" | "out" ) ] [ "on" ifspec ]
                    [ af ] [ protospec ] [ hosts ] [ filteropt-list ] [ "{"
]
```

```
   anchor-close   = "}"
```

```
   trans-anchors  = ( "nat-anchor" | "rdr-anchor" | "binat-anchor" ) string
                    [ "on" ifspec ] [ af ] [ "proto" ] [ protospec ] [ hosts
]
```

```
   load-anchor    = "load anchor" string "from" filename
```

```
   queueopts-list = queueopts-list queueopts | queueopts
   queueopts      = [ "bandwidth" bandwidth-spec ] |
                    [ "qlimit" number ] | [ "tbrsize" number ] |
                    [ "priority" number ] | [ schedulers ]
   schedulers     = ( cbq-def | priq-def | hfsc-def )
   bandwidth-spec = "number" ( "b" | "Kb" | "Mb" | "Gb" | "%" )
```

```
   action         = "pass" | "block" [ return ] | [ "no" ] "scrub"
   return         = "drop" | "return" | "return-rst" [ "( ttl" number ")" ]
|
                    "return-icmp" [ "(" icmpcode [ [ "," ] icmp6code ] ")" ]
|
                    "return-icmp6" [ "(" icmp6code ")" ]
   icmpcode       = ( icmp-code-name | icmp-code-number )
   icmp6code      = ( icmp6-code-name | icmp6-code-number )
```

```
   ifspec         = ( [ "!" ] ( interface-name | interface-group ) ) |
                    "{" interface-list "}"
   interface-list = [ "!" ] ( interface-name | interface-group )
                    [ [ "," ] interface-list ]
   route          = ( "route-to" | "reply-to" | "dup-to" )
                    ( routehost | "{" routehost-list "}" )
                    [ pooltype ]
   af             = "inet" | "inet6"
```

```
   protospec      = "proto" ( proto-name | proto-number |
                    "{" proto-list "}" )
   proto-list     = ( proto-name | proto-number ) [ [ "," ] proto-list ]
```

```
   hosts          = "all" |
                    "from" ( "any" | "no-route" | "urpf-failed" | "self" |
host |
                    "{" host-list "}" | "route" string ) [ port ] [ os ]
                    "to"   ( "any" | "no-route" | "self" | host |
```

```
                    "{" host-list "}" | "route" string ) [ port ]
```

```
   ipspec         = "any" | host | "{" host-list "}"
   host           = [ "!" ] ( address [ "/" mask-bits ] | "<" string ">" )
   redirhost      = address [ "/" mask-bits ]
   routehost      = "(" interface-name [ address [ "/" mask-bits ] ] ")"
   address        = ( interface-name | interface-group |
                    "(" ( interface-name | interface-group ) ")" |
                    hostname | ipv4-dotted-quad | ipv6-coloned-hex )
   host-list      = host [ [ "," ] host-list ]
   redirhost-list = redirhost [ [ "," ] redirhost-list ]
   routehost-list = routehost [ [ "," ] routehost-list ]
```

```
   port           = "port" ( unary-op | binary-op | "{" op-list "}" )
   portspec       = "port" ( number | name ) [ ":" ( "*" | number | name ) ]
   os             = "os"  ( os-name | "{" os-list "}" )
   user           = "user" ( unary-op | binary-op | "{" op-list "}" )
   group          = "group" ( unary-op | binary-op | "{" op-list "}" )
```

```
   unary-op       = [ "=" | "!=" | "<" | "<=" | ">" | ">=" ]
                    ( name | number )
   binary-op      = number ( "<>" | "><" | ":" ) number
   op-list        = ( unary-op | binary-op ) [ [ "," ] op-list ]
```

```
   os-name        = operating-system-name
   os-list        = os-name [ [ "," ] os-list ]
```

```
   flags          = "flags" ( [ flag-set ] "/"  flag-set | "any" )
   flag-set       = [ "F" ] [ "S" ] [ "R" ] [ "P" ] [ "A" ] [ "U" ] [ "E" ]
                    [ "W" ]
```

```
   icmp-type      = "icmp-type" ( icmp-type-code | "{" icmp-list "}" )
   icmp6-type     = "icmp6-type" ( icmp-type-code | "{" icmp-list "}" )
   icmp-type-code = ( icmp-type-name | icmp-type-number )
                    [ "code" ( icmp-code-name | icmp-code-number ) ]
   icmp-list      = icmp-type-code [ [ "," ] icmp-list ]
```

```
   tos            = ( "lowdelay" | "throughput" | "reliability" |
                    [ "0x" ] number )
```

```
   state-opts     = state-opt [ [ "," ] state-opts ]
   state-opt      = ( "max" number | "no-sync" | timeout | "sloppy" |
"pflow" |
                    "source-track" [ ( "rule" | "global" ) ] |
                    "max-src-nodes" number | "max-src-states" number |
                    "max-src-conn" number |
                    "max-src-conn-rate" number "/" number |
                    "overload" "<" string ">" [ "flush" ] |
                    "if-bound" | "floating" )
```

```
fragmentation  = [ "fragment reassemble" | "fragment crop" |
                 "fragment drop-ovl" ]
```

```
timeout-list   = timeout [ [ "," ] timeout-list ]
timeout        = ( "tcp.first" | "tcp.opening" | "tcp.established" |
                 "tcp.closing" | "tcp.finwait" | "tcp.closed" |
                 "udp.first" | "udp.single" | "udp.multiple" |
                 "icmp.first" | "icmp.error" |
                 "other.first" | "other.single" | "other.multiple" |
                 "frag" | "interval" | "src.track" |
                 "adaptive.start" | "adaptive.end" ) number
```

```
limit-list     = limit-item [ [ "," ] limit-list ]
limit-item     = ( "states" | "frags" | "src-nodes" ) number
```

```
pooltype       = ( "bitmask" | "random" |
                 "source-hash" [ ( hex-key | string-key ) ] |
                 "round-robin" ) [ sticky-address ]
```

```
subqueue       = string | "{" queue-list "}"
queue-list     = string [ [ "," ] string ]
cbq-def        = "cbq" [ "(" cbq-opt [ [ "," ] cbq-opt ] ")" ]
priq-def       = "priq" [ "(" priq-opt [ [ "," ] priq-opt ] ")" ]
hfsc-def       = "hfsc" [ "(" hfsc-opt [ [ "," ] hfsc-opt ] ")" ]
cbq-opt        = ( "default" | "borrow" | "red" | "ecn" | "rio" )
priq-opt       = ( "default" | "red" | "ecn" | "rio" )
hfsc-opt       = ( "default" | "red" | "ecn" | "rio" |
                 linkshare-sc | realtime-sc | upperlimit-sc )
linkshare-sc   = "linkshare" sc-spec
realtime-sc    = "realtime" sc-spec
upperlimit-sc  = "upperlimit" sc-spec
sc-spec        = ( bandwidth-spec |
                 "(" bandwidth-spec number bandwidth-spec ")" )
include        = "include" filename
```

FICHIERS

- /etc/hosts Base de données des noms d'hôte.
- /etc/pf.conf Endroit par défaut du fichier de règles.
- /etc/pf.os Endroit par défaut des empreintes d'OS.
- /etc/protocols Base de données des noms de protocoles.
- /etc/services Base de données des noms de services.

VOIR AUSSI

carp(4), icmp(4), icmp6(4), ip(4), ip6(4), pf(4), pflow(4), pfsync(4), route(4), tcp(4), udp(4), hosts(5),

pf.os(5), protocols(5), services(5), ftp-proxy(8), pfctl(8), pflogd(8), route(8)

HISTORIQUE

Le format de fichier fichier **pf.conf** est apparu dans OpenBSD 3.0.

OpenBSD 4.5 31 janvier 2009 33

From:
https://www.mouet-mouet.net/ - **mouet-mouet !**

Permanent link:
https://www.mouet-mouet.net/doku.php?id=maxime:openbsd:manpages-fr:5:pf.conf

Last update: **2021/10/08 00:17**