# Configuring IPv6 Firewalls with pf

János Mohácsi
NIIF/HUNGARNET

# What is pf?

- OpenBSD included IPFilter in the default install since 3.0. Included in FreeBSD since 5.3 (5.x as port) and in NetBSD since 2.0.
- Ideas from ipf which is available for Linux, Solaris, HP-UX, IRIX additionally to the operating systems above.
- Principles
  - working on IP packet level (vs. application level proxies or ethernet level bridges)
  - intercepting each IP packet that passes through the kernel (in and out on each interface), passing or blocking it
  - stateless inspection based on fields of each packet
  - stateful filtering keeping track of connections, additional information makes filtering more powerful (sequence number checks) and easier (replies, random client ports)
  - filtering for local host or network (multihomed host, IP forwarding or bridging)

# pf filter rules

- linear linked list, evaluated top to bottom for each packet (unlike netfilter's chains tree)
- rules contain parameters that match/mismatch a packet rules pass or block a packet
- last matching rule wins (except for 'quick', which aborts rule evaluation)
- rules can create state, further state matching packets are passed without rule set evaluation

# common basic rule syntax

```
pass|block in|out on <int> [<af>] [proto
  <protocol>] from <src_ip> port <src_port>
  to <dst_ip> port <dst_port>
```

<int>: a network interface

<af>: address family: inet or inet6

<protocol: protocol: icmp, tcp, udp, icmp6 etc.

<src_ip>: an IP address (or range)

<src_port>: a TCP or UDP port number

<dst_ip>: an IP address (or range)

<dst_port>: a TCP or UDP port number

| denotes OR, [] denotes optional part

## common basic rule syntax/2

```
pass|block in|out on all
```

- apply a rule to all interfaces, or to all sources, and to all destinations, and to all ports the powerful all keyword should be employed:

## Small examples

1. block out on tun0 all port 25
2. pass in on fxp0 from 192.168.1.0/24 to 192.168.2.51 port 22
3. pass out on fxp0 from 192.168.2.51 to 192.168.1.0/24 port 22

## Common basic rule syntax /3

- Quick
  - Using the quick keyword in a rule commands PF to apply the rule immediately. Further rule processing is abandoned.
- Log
  - The all important logging feature is now in use via the log keyword. Logging is accomplished emitting bpf like log via virtual network inteface pflog0 that can be collected and filtered by by the pflogd daemon. Logging can be applied to any rule.
    - link layer header used for pf related information (rule, action)
    - Easy to process with tcpdump
    - Important: when using log and state keywords in the same rule: only the packet that establishes the state is logged.

# State table

- pf can "keep state" or perform "stateful inspection". Stateful inspection refers to PF's ability to track the state, or progress, of a network connection. By storing information about each connection in a state table, PF is able to determine if a packet passing through the firewall belongs to an already established connection. If it does, it is passed through the firewall without going through any ruleset evaluation.
  - TCP (sequence number checks on each packet), ICMP error messages match referred to packet (simplifies rules without breaking PMTU discoveries etc.)
  - UDP, ICMP queries/replies, other protocols: pseudo-connections with timeouts
  - adjustable timeouts (aggressive, normal, high-latency, conservative )
  - binary search tree (AVL, now Red-Black), O(log n) even in worst-case
  - key is two address/port pairs

# Ruleset IPv4 1(nothing in, DNS, all TCP, ping out)

```
EXT = “bge0"
LAN = “bge1"
LANip4 = "192.168.1.1“
EXTip4 = “192.168.2.1
LANnet4 = "192.168.1.0/24“
Lo4 = “127.0.0.1”
# expire state connections early
set optimization aggressive
block in log all
# allow DNS requests to go out
pass out on $EXT inet proto udp from {$EXTip4, $Lo4, $LANnet4} to any port=domain keep state
# all TCP request allowed out
pass out on $EXT inet proto tcp from {EXTip4, $Lo4, $LANnet4} to any keep state
# all ping request allowed out
pass out on $EXT inet proto icmp all icmp-type 8 code 0 keep state
# DNS request inside
pass in on $LAN inet proto from $LANnet4 to any port domain
# TCP request inside
pass in on $LAN inet proto tcp from $LANnet4 to any
# ICMP request inside
pass in on $LAN inet proto icmp all icmp-type 8 code 0
```

# Antispoofing

- antispoof, parser generates blocking rules appropriate for the specified interfaces

```
antispoof for lo0
   block in on !lo0 inet from 127.0.0.1/8 to any
   block in on !lo0 inet6 from ::1 to any
antispoof for bge1 #inet/inet6
   block in on !bge1 inet from 192.168.1.1/24 to any
   block in inet from 192.168.1.1 to any
   block in on !bge1 inet6 from 2001:db8:1:2::/64 to any
   block in on !bge1 inet6 from fe80::209:6bff:fe8c:845b to
     any
   block in inet6 from 2001:db8:1:2::1 to any
```

# IPv6 specific rules -reminder

- Neighbor solicitation/neighbor advertisement is REQUIRED – icmp-type 135/136 (ipv6-icmp-type neighbrsol/neighbradv)
- For stateless address autoconfiguration router advertisement and router solicitation REQUIRED– icmp type 133/134 (ipv6-icmp-type routersol/routeradv)
- Path MTU discovery – automatic if you use keep-state
  - Same applies for Destination unreachable, Time exceeded and IPv6 Parameter problem messages
  - Otherwise you have to build your own rules

# Supported ICMPv6 types

| | | |
|---|---|---|
| unreach | 1 | Destination unreachable |
| toobig | 2 | Packet too big |
| timex | 3 | Time Exceeded |
| paramprob | 4 | Parameter problem |
| echoreq | 128 | Echo Request |
| echorep | 129 | Echo Reply |
| groupqry | 130 | ICMPv6 Membership query |
| listqry | 130 | MLD listener query |
| grouprep | 131 | ICMPv6 membership report |
| listenrep | 131 | MLD listener report |
| groupterm | 132 | ICMPv6 membership termination |
| listendone | 132 | MLD listener done |
| routersol | 133 | ND router solicitation |
| routeradv | 134 | ND router advertisement |
| neighbrsol | 135 | ND neighbor solicitation |
| neighbradv | 136 | ND neighbor advertisement |

# Supported ICMPv6 types /2

| | | |
|---|---|---|
| redir | 137 | ND redirection |
| routerrenu m | 138 | ICMPv6 router renumbering |
| wrureq | 139 | Who are you request |
| wrurep | 140 | Who are you reply |
| fqdnreq | 139 | ICMPv6 Fully Qualified Domain Name Query |
| fqdnrep | 140 | ICMPv6 Fully Qualified Domain Name Reply |
| nireq | 139 | Neighbor Information Query |
| nirep | 140 | Neighbor Information Reply |
| mtraceres p | 200 | MLD Multicast trace response |
| mtrace | 201 | MLD Multicast trace |

## Ruleset IPv6 1(nothing in, DNS, all TCP, ping out)

```
EXT = “bge0"
LAN = “bge1"
LANip6 = “2001:db8:1:1::1“
EXTip6 = “2001:db8:1:2::1”
LANnet6 = “2001:db8:1:1::1/64“
Lo6 = “::1”
# expire state connections early
set optimization aggressive
block in log all
# allow DNS requests to go out
pass out on $EXT inet6 proto udp from {$EXTip6, $Lo6, $LANnet6} to any port=domain keep
   state
# all TCP request allowed out
pass out on $EXT inet6 proto tcp from {EXTip6, $Lo6, $LANnet6} to any keep state
# all ping request allowed out
pass out on $EXT inet6 proto icmp6 all icmp6-type echoreq keep state
# ND solicitation out
pass out on $EXT inet6 proto icmp6 all icmp6-type {neighbradv, neighbrsol}
# ND advertisement in
pass in on $EXT inet6 proto icmp6 all icmp6-type {neighbradv, neighbrsol}
```

## Ruleset IPv6 1(continue – with router advertisement from FW)

```
#router advertisement out
pass out on $LAN inet6 proto icmp6 all icmp6-type routersadv
# router solicitation in
pass in on $LAN inet6 proto icmp6 all icmp6-type routerrsol
# DNS request inside
pass in on $LAN inet6 proto from $LANnet6 to any port domain
# TCP request inside
pass in on $LAN inet6 proto tcp from $LANnet6 to any
# ICMP request inside
pass in on $LAN inet6 proto icmp6 all icmp6-type echoreq
```

## Ruleset IPv6 2 (allow access to internal mail & www server – additional rules)

```
#internal server address
LANSRV6="2001:db8:1:2::2"
LANSRV4="192.168.1.2"
#allow incoming connection to SMTP server
pass in on $EXT inet6 proto tcp from any to $LANSRV6 port=25 keep-state
pass in on $EXT inet proto tcp from any to $LANSRV4 port=25 keep-state
#all reply from SMTP server (does not really necessary)
pass in on $LAN inet6 proto tcp from $LANSRV6 port=25 to any keep-state
pass in on $LAN inet proto tcp from $LANSRV4 port=25 to any keep-state
#allow incoming connection to WWW server
pass in on $EXT inet6 proto tcp from any to $LANSRV6 port=www keep-state
pass in on $EXT inet proto tcp from any to $LANSRV4 port=www keep-state
#all reply from SMTP server (does not really necessary)
pass in on $LAN inet6 proto tcp from $LANSRV6 port=www to any keep-state
pass in on $LAN inet proto tcp from $LANSRV4 port=www to any keep-state
```

## Problems with IPv6

- No fragment normalisation – not possible! – fragmentation only at the end-host
- no real support for extension headers – check existence of extension header possible without chain processing
- IPv6 fragments are blocked unconditionally
- No IPv6 support in ftp-proxy
- No support to filter inside tunnel – except if tunnel terminated at firewall

# Tunnel filtering example - simplest

```
pass out on $EXT inet proto ipv6 from $EXT
  to $TUNNELREMOTE4 keep state
pass in  on $EXT inet proto ipv6 from
  $TUNNELREMOTE4 to $ext_if  keep state
pass out on gif0 inet6 all keep state
pass in  on gif0 inet6 all keep state
```

# More restrictive IPv6 rules

- Allow Rtsol/rtadv on a more specific address

```
pass out on $LAN inet6 proto ipv6-icmp from
  fe80::/16 to ff02::2 icmp6-type routersol code 0
pass in  on $LAN inet6 proto ipv6-icmp from
  fe80::/16 to ff02::1 icmp6-type routeradv code 0
pass in  on $LAN inet6 proto ipv6-icmp from
  fe80::/16 to fe80::/16 icmp6-type routeradv code
  0
```

- Allow NDsol/Ndadv on more specific address

```
pass out on $if inet6 proto ipv6-icmp from { ::
  fe80::/16 } to ff02::/16 icmp6-type grouprep code
  0
pass out on $if inet6 proto ipv6-icmp from ($if) to
  any icmp6-type neighbrsol code 0
pass in  on $if inet6 proto ipv6-icmp from any to
  ($if) icmp6-type neighbradv code 0
```

# End

- Further information
  - The OpenBSD pf FAQ:
    - http://www.openbsd.org/faq/pf/index.html
  - Tons of information:
    - http://www.benzedrine.cz/pf.html
    - mailing list with archive