Firewalling with OpenBSD's «pf»

Stephan A. Rickauer
StarTek – secure by design,
based on Peter Hansteens Tutorial

Universität Konstanz, 13.07.2007

Introduction

Default packet filter from OpenBSD 3.0 (Dec. 2001)

- Replaced IPFilter due to licensing issues
- Based on new code by Daniel Hartmeier
- Good performance, low maintenance
- Fully ipv6 compatible
- In the base systems of
  - OpenBSD
  - FreeBSD
  - NetBSD
  - DragonFlyBSD

Packet Filter?

- Kernel level code
- Configuration file (`/etc/pf.conf`)
- User space control tool (`/sbin/pfctl`)
- Man pages!

- modify, drop or pass packets (“decisions”)
  - “Stopping undesirable traffic”
  - Network address translation (NAT)
  - Traffic normalisation, shaping ...

Seven Type of Statements

1. Macros
2. Tables
3. Options
4. Traffic Normalization (“scrubbing”)
5. Queueing
6. Translation
7. Packet Filtering

Macros

- User-defined variables
- Expanded later in context
- Defined at the very top of pf.conf

```
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

- Named structures
- for large numbers of addresses
- Faster than a large number of similar rules

```
table <private> const { 10/8, 172.16/12, 192.168/16 }
table <badhosts> persist
block on fxp0 from { <private>, <badhosts> } to any
```

Options

- tune the behaviour of the packet filtering engine

```
set limit states 20000
set block-policy return
set timeout tcp.first 120
set optimization satellite
.
.
.
```

Traffic Normalisation (“scrubbing”)

- sanitizes packet content
  - Remedies ambiguities
  - IP fragment reassembly

```
scrub in on $ext_if fragment reassemble min-ttl
```

Queueing

- Bandwidth control
- Three queue types
  - Priority Queueing (priq)
    - Flat
    - Unique priority per queue (1-15)
  - Hierarchical Fair Service Curve (hfsc)
    - service curve based QoS model
    - Decouple delay and bandwidth allocation

Queueing & Example

- Class Based Queueing (cbq)
  - Hierachical tree with child queues
  - Each queue gets priority and bandwidth assigned
  - Queues can borrow bandwidth from parent

```
altq on dc0 cbq bandwidth 5Mb queue { std, http, mail, ssh }
queue std bandwidth 10% cbq(default)
queue http bandwidth 60% priority 2 cbq(borrow) {employ,devel}
queue   devel bandwidth 75% cbq(borrow)
queue   employ bandwidth 15%
queue mail bandwidth 10% priority 0 cbq(borrow)
queue ssh bandwidth 20% cbq(borrow) { ssh_interactive, ssh_bulk }
queue   ssh_interactive bandwidth 50% priority 7 cbq(borrow)
queue   ssh_bulk bandwidth 50% priority 0 cbq(borrow)
```

Fun with ALTQ

```
altq on $ext_if cbq queue { q_default q_web q_mail }

    queue q_default cbq(default)
    queue q_web (...)

     ## all mail limited to 1Mb/sec
     queue q_mail bandwidth 1Mb { q_mail_windows }
     ## windows mail limited to 56Kb/sec
     queue q_mail_windows bandwidth 56Kb
```

" I can't believe I didn't see this earlier. Oh, how sweet. ... Already a huge difference in my load. Bwa ha ha."

Randal L. Schwartz

Translation

- modify either the source or destination address
- Stateful
- Three types of translation
  - binat: bidirectional mapping
  - nat: network address translation
  - rdr: redirect (IP and port)

```
rdr on $ext_if proto tcp from any to any \
   port 80 -> 127.0.0.1 port 8080
nat on $ext_if inet from ! ($ext_if) \
   to any -> ($ext_if)
```

Translation, more examples

```
binat on $ext_if from 10.1.2.150 to any -> $ext_if

nat on $ext_if inet from any to any -> 192.0.2.16/28
  source-hash

rdr on $ext_if proto tcp from any to any port 80 \
   -> { 10.1.2.155, 10.1.2.160, 10.1.2.161 } \
   round-robin
```

Packet Filtering

- block and pass packets
- based on attributes of layer 3 and layer 4
- sequential order

(Examples coming ...)

Take a breath ...

Starting pf

/etc/rc.conf.local

```
pf=YES                  # enable  pf
pf_rules=/etc/pf.conf # specify file containing rules
```

enable with

```
$ sudo pfctl -e && sudo pfctl -f /etc/pf.conf
```

Note: Rebooting loads the rc script's default rule set

Simple setup – single machine

/etc/pf.conf

```
block in all
pass out all keep state


# OpenBSD 4.1 keeps state by default
block in all
pass out all
```

Slightly stricter

/etc/pf.conf

```
tcp_services = "{ ssh, smtp, domain, www, pop3s }"
udp_services = "{ domain }"

block all
pass out proto tcp to any port $tcp_services
pass out proto udp to any port $udp_services
```

for syntax check only:

```
$ sudo pfctl -nf /etc/pf.conf
```

Statistics

```
$ sudo pfctl -s info
Status: Enabled for 17 days 00:24:58           Debug: Urgent
Interface Stats for ep0                  IPv4              IPv6
  Bytes In                          9257508558                 0
  Bytes Out                          551145119               352
  Packets In
    Passed                             7004355                 0
    Blocked                              18975                 0
  Packets Out
    Passed                             5222502                 3
    Blocked                                 65                 2

State Table                             Total              Rate
  current entries                          15
  searches                             19620603            13.3/s
  inserts                                173104             0.1/s
  removals                               173089             0.1/s ...
```

More Statistics

```
...
Counters
  match                                196723                 0.1/s
  bad-offset                                0                 0.0/s
  fragment                                 22                 0.0/s
  short                                     0                 0.0/s
  normalize                                 0                 0.0/s
  memory                                    0                 0.0/s
  bad-timestamp                             0                 0.0/s
  congestion                                0                 0.0/s
  ip-option                                28                 0.0/s
  proto-cksum                             325                 0.0/s
  state-mismatch                          983                 0.0/s
  state-insert                              0                 0.0/s
  state-limit                               0                 0.0/s
  src-limit                                26                 0.0/s
  synproxy                                  0                 0.0/s
```

Pitfalls: in, out, on

If you write

```
pass in on ep1 from ep1:network to ep0:network \
     port $ports keep state
```

then you also need

```
pass out on ep0 from ep1:network to ep0:network \
     port $ports keep state
```

but do you actually mean

```
pass from ep1:network to any port $ports keep state
```

ftp-proxy

/etc/rc.conf.local: ftpproxy_flags=""

/etc/pf.conf:

```
#NAT section anchors
nat-anchor "ftp-proxy/*"
rdr-anchor "ftp-proxy/*"

#the redirection
rdr pass on $int_if proto tcp from any to any \
    port ftp -> 127.0.0.1 port 8021

#filtering section
anchor "ftp-proxy/*"
pass out proto tcp from $proxy to any port 21
```

Making your network troubleshooting friendly

- Mainly concerns the Internet Control Message Protocol (ICMP)
  - Control Messages (transmission parameters, packet sizes, routing, path MTU discovery)
  - Mid 1990s Ping of Death scare - ICMP 'just evil'
  - Modern OSes not vulnerable, but people are still scared

```
pass inet proto icmp from any to any
```

Pro: makes debugging easier

Con: may reveal too much about your network

Better:

```
icmp_types = "echoreq"
pass inet proto icmp all icmp-type $icmp_types keep state
```

Helping traceroute

traceroute needs a bit of help, but

uses a fixed formula:

```
# allow out the default range for traceroute(8):
# "base+nhops*nqueries-1" (33434+64*3-1)
pass out on $ext_if inet proto udp from any to \
  any port 33433 >< 33626 keep state
```

Note: Unix traceroute uses UDP by default;

Microsoft uses ICMP ECHO (like unix with -I)

Network Hygiene

- Block-policy:

```
set block-policy return
```

- Scrub:

```
scrub in all
```

- Antispoof ("this packet should not be here")

```
antispoof for $ext_if
antispoof for $int_if
```

Keyword "log" in the rules to be logged

```
set loginterface $ext_if
pass out log from <client> to any port $email
  label client-email keep state
```

- Logs in binary, tcpdump(8) readable format
- label creates counters for statistics
- logs only initial packet, use `log (all)` to log all matching packets

pftop

```
pfTop: Up State 1-21/67, View: default, Order: none, Cache: 10000       19:52:28

PR   DIR SRC                  DEST                  STATE   AGE   EXP  PKTS BYTES
tcp  Out 194.54.103.89:3847   216.193.211.2:25       9:9     28    67    29  3608
tcp  In  207.182.140.5:44870  127.0.0.1:8025         4:4     15 86400    30  1594
tcp  In  207.182.140.5:36469  127.0.0.1:8025        10:10   418    75   810 44675
tcp  In  194.54.107.19:51593  194.54.103.65:22       4:4    146 86395   158 37326
tcp  In  194.54.107.19:64926  194.54.103.65:22       4:4    193 86243   131 21186
tcp  In  194.54.103.76:3010   64.136.25.171:80       9:9    154    59    11  1570
tcp  In  194.54.103.76:3013   64.136.25.171:80       4:4      4 86397     6  1370
tcp  In  194.54.103.66:3847   216.193.211.2:25       9:9     28    67    29  3608
tcp  Out 194.54.103.76:3009   64.136.25.171:80       9:9    214     0     9  1490
tcp  Out 194.54.103.76:3010   64.136.25.171:80       4:4     64 86337     7  1410
udp  Out 194.54.107.18:41423  194.54.96.9:53         2:1     36     0     2   235
udp  In  194.54.107.19:58732  194.54.103.66:53       1:2     36     0     2   219
udp  In  194.54.107.19:54402  194.54.103.66:53       1:2     36     0     2   255
udp  In  194.54.107.19:54681  194.54.103.66:53       1:2     36     0     2   271
```

Invisible gateway - bridge

- Bridge: machine with no IP address of its own, between the Internet and a local network
  - Opererates on the Ethernet level
  - "Invisible" to the outside world
  - Is able to use PF for filtering and nat/rdr

Invisible gateway - bridge

/etc/hostname.ep0: up

/etc/hostname.ep1: up

/etc/bridgename.bridge0:

```
   add ep0 add ep1 blocknonip ep0 blocknonip ep1 up
```

```
ext_if = ep0
int_if = ep1
interesting-traffic = { ... }
block all
pass quick on $ext_if all
pass log on $int_if from $internal_net to any port \
  $interesting-traffic keep state
```

pf Integration

- authpf
  - Non-interactive shell
  - After authentication, user's IP is added to `<authpf_users>`
  - Table gets destroyed when ssh session terminates
- dhcpd
  - Control “camping”
- Ftp-proxy
  - Dynamically allow outgoing & incoming ftp connections
- hoststated
  - Check remote host status & change tables accordingly
- Spamd
  - Greylisting, greytrapping, blacklisting

Turning away the brutes

```
table <bruteforce> persist

block quick from <bruteforce>

pass inet proto tcp from any to $int_if:network
  port $tcp_services \
  (max-src-conn 100, max-src-conn-rate 15/5, \
  overload <bruteforce> flush global)
  ...
```

or

- put overloaders in a minimal-bandwidth queue (ALTQ)
- rdr overloaders to specific site

Support OpenBSD!

- If you enjoyed this: Support OpenBSD.
- Buy OpenBSD CDs and other items, donate.
- This is no joke.

- Remember: Developing and maintaining Free Software takes real work and real money.

- OpenBSD Consulting:
  StarTek – secure by design