SANE 2006: OpenBSD Packet Filter

15/05/06 11:03

PF = default packet filter in OpenBSD since 2001, after IPFilter was found to contain unlicensed pieces of code. For a reference re: performance -> http://www.benzedrine.cx/pf-paper.html

Packet filter = kernel level code (module + admin software) which determines the path for network packets. Naturally it can block undesired traffic, thus becoming a form of firewall.

PF also does NAT which is basically a quick fix for the problem IPv6 was supposed to fix, mainly the shortage of IP addresses.

Current status of PF includes:

- Filtering on protocol, port, packete type, address, operating system
- Redirection (to daemons or machines)
- NAT
- Load balancing and traffic shaping
- "Human readable" configuration -> no shell scripts with many switches and parameters
- It is part of the base system for OpenBSD, FreeBSD, NetBSD and DragonFlyBSD.

Configuration of PF is done using **`/etc/pf.conf`** or **`pfctl`**.

Basic configuration in OpenBSD:

- **`/etc/rc.conf`**
  - **`pf=YES`**
  - **`pf_rules=/etc/pf.conf`**
- **`sudo pfctl –e; sudo pfctl –f /etc/pf.conf`**
- Default ruleset zorgt dat DNS, SSH en NFS altijd werken. Dit om te voorkomen dat je jezelf buiten sluit. Deze word geladen als **`pf.conf`** leeg is, of niet goed ingericht. Netjes!

Basic configuration in FreeBSD:

- **`/etc/rc.conf`**
  - **`pf_enable=YES`**
  - **`pf_rules=/etc/pf.conf`**
  - **`pf_flags=""`**

    - **pflog_enable=YES**
    - **pflog_logfile="/var/log/pflog"**
    - **pflog_flags=""**

Basic configuration for NetBSD

- Kernel config file
    - **pseudo-device pf**
    - **pseudo-device pflog**
- **/etc/rc.conf**
    - **pf=YES**
    - **pflog=YES**

First rule set – single machine

- **/etc/pf.conf**
    - **block is ALL**
    - **pass out ALL keep state**
- Enable with
    - **sudo pfctl –e; sudo pfctl –f /etc/pf.conf**

Nice thing is that you can use macros. For example: /etc/services is used, so you can use service names as well. For example:

```
top_services = "{ ssh, smtp, domain, www }"
udp_services = "{ domain }"
block all
pass out proto tcp to any port $tcp_services keep state
pass proto udp to any port $udp_services keep state
```

Then reload using **pfctl –f** or do a syntax check using **pfctl –nf**.

You can gather statistics from PF, using **pfctl –s info**.

When running as a gateway system, IN and OUT refer to the gateway system itself and not to your "internal network". PF has no notion which network is which.

Examples:

```
pass in on x11 from x11:network to x10:network port $ports
keep state
```

pass out on x10 from x11:network to x10:network port $ports keep state

You can define a macro for your local network:

```
$int_if = x11
localnet = $int_if:network
pass from $localnet to any port $ports keep state
```

So just like in programming, the use of macros (variables) is a good thing!

To enable IP Forwarding in BSD:

- **sysctl net.inet.ip.forwarding=1**
- **sysctl net.inet6.ip6.forwarding=1**
- **/etc/sysctl.conf (OpenBSD)**
  - **set.inet-ip-forwarding=1, or**
- **/etc/rc.conf (FreeBSD)**
  - **gateway_enable="YES"**
  - **ipv5_gateway_enable="YES"**

Simple gateway with NAT

- **nat on $ext_if from $localnet to any -> ($ext_if)**
- **pass from (le0, $int_if:network) to any keep state**

WEIRD:

- The config file is read from top to bottom,
- BUT: the last rule found is applied!
- Unless you use the keyword **pass quick** (which forces the rule to be used).

FTP is and will also be a problem with packet filters. Clear text passwords, more than one connection and it uses random ports. We recommend you use something completely different :) If you can't, here's what you do using redirection.

- Reconfigure **/etc/inetd.conf** to use **127.0.0.1#021 /usr/lobexec/ftp-proxy –n**
- **rdr on $int_if proto tcp from any to any port ftp -> 127.0.0.1 port $021**

- **pass in on $ext_if inet proto tcp from port ftp-data to ($ext_if_ user proxy flags s/SA keep state**
- Look up FTPSESAME for a better solution.
- Or even beter, use the newer version ftp-proxy.
    - This does not use inetd, but is started from **rc.conf**
    - You will need an anchor for ftp-proxy.
    - Will generate rules on the fly for active FTP sessions.

In the past ICMP messages were mostly blocked by all firewalls, due to the *Ping Of Death* scare. Modern OSes are mostly resistant against these attacks, so you really don't need to block all of it. This will make troubleshooting your networks much easier. However, be sure to not allow everything, since outsiders will be able to discover your whole networks! :3 A good solution:

- **icmp_types = "echoreq" # PING :)**
- **pass inet proto icmp icmp-type $icmp_types from $int_if:network to any keep state**
- **pass inet proto icmp icmp-type $icmp_types from any to $ext_if keep state**
- Traceroute needs a little bit more help -> **pass out on $ext_if inet proto udp from any to any port 33433 >< 44626 keep state**

Blocking policies:

- return = the default where a signal is sent back "Connection refused. Destination unreachable", etc.
- drop = drop without return code
- **set block-policy return**

Scrubbing:

- normalization and defragmentation of packets, without much of a performance hit.
- **scrub is all**

Anti spoofing:

- **antispoof for $ext_if**
- **antispoof for $int_if**

Blocking non-routable traffic

- **martians = "{ 127.0.0.0/8, 192.16.0.0/16 }" # and so on**
- **block drop in quick on $ext_if from $martians to any**
- **block drop out quick on $ext_if from any to $martians**

Webserver and mail server on the inside

- **webserver = "192.168.2.7"**
- **webports = "{ https, https }"**
- **emailserver = "182.168.2.5"**
- **email = "{ smtp, pop3, imap, imap4, imaps, pop3s }"**
- **rdr on $ext_if proto tcp from any to $ext_if port #webports -> $webserver**
- **rdr on $ext_if proto tcp from any to $ext_if port $email -> $emailserver**
- **pass in on $ext_if prot tcp from any to $webserver port #webports flags S/SA synproxy state**
- **pass in on $ext_if proto tcp from any to $emailserver port $email flasg S/SA synproxy state**
- **pass out on $ext_if prot tcp from $emailserver to any port smtp flags S/SA synproxy state.**
- PROBLEM: traffic from the inside does not reach the interface interface.
- SOLUTION: split horizon DNS, proxying, moving servers to DMZ. Or add the following:
  - **rdr on $int_if proto tcp from $int_if:network to $ext_if port $webports -> $webserver**
  - **rdr on $int_if proto tcp from $int_if:network to $ext_if port $email -> $emailserver**
  - **no nat on $int_if proto tcp from $int_if ....** FSCK HE CHANGED TO THE NEXT PAGE :(

Tables are supposed to make your life easier.

table <clients> { 192.168.02.0/24, !192.168.2.5 }

- Or in: **/etc/clients**
  - **192.168.2.0/25**
  - **!192.168.2.5**
- Then in: **/etc/pf.conf**
  - **table <clients> persist file /etc/clients**

        - **pass inet proto tcp from <clients> to any port $client_out flags S/SA keep state**

Add a table entry (all of which can be added to cron for automatic lock-down):

sudo pfctl –t clients –T add 192.168.1.0/16 (only the in-memory copy)
sudo pfctl –t clients –T show >/etc/clients
sudo pfctl –t clients –T replace –f /etc/clients

In order to log traffic, just add the keyword "log" to the rules you want logged. Traffic is stored in binary, TCPDump-readable format. You can also add the keyword "label" to create counters for statistics. But please, try not to log too much, since the files will grow incredibly large if you have a lot of traffic.

pftop = top for packet filter :) yay!

You can also make your gateway invisible, by running in bridge mode.

- **/etc/hostname.x10**
    - **up**
- **/etc/hostname.x11**
    - **up**
- **/etc/bridgename.bridge0**
    - **add x10 add x11 blocknonip x10 blocknonip x11 up**
- **/etc/pf.conf**
    - **ext_if = x10**
    - **int_if = x11**
    - **traffic = { ... }**
    - **block all**
    - **pass quick on $extern all**
    - **pass log on $int_if from $internal to any port *traffic keep state**
- see also brdige(4) and brconfig(8)

Alternate queueing = altq = queues for bandwidth allocation and traffic shaping.

- class based CBQ (percent, bytes), or

- priority base PRIQ, or
- hierarchical ???
- www.benzedrine.cx/ackpri.html
    - **altq on $ext_if priq bandwidth 100Kb queue { q_pri, q_def }**
    - **queue q_pri priority 7**
    - **queue q_def priority 1 priq(default)**
    - **pass out on $ext_if proto tcp from $ext_if to any flags S/SA keep state queue (q_def, q_pri)**
    - **pass in on $ext_if proto tcp from any to $ext_if flags S/SA keep state queue (q_def, q_pri_**
- www.unix.se
    - **queue def bandwidth 18% cgq(default borrow red)**
    - **queue ftp bandwidth 10% cgq(borrow red)**
    - and so on...

CARP

- "Free" alternative to VRRP (virtual router redundancy protocol) -> a fail-over mechanism for cisco, ibm and nokia equipment.
- You set up virtual network interfaces for automatic fail-over.

PFsync

- virtual network interface
- handles synchronisation between pf firewalls
- www.countersiege.com/doc/pfsync-carp

Authpf

- Can be used to set up security for a wireless gateway. This can be used to set up a standard web interface which describes how to get access to the network (credit cards and such).
- **/etc/authpf/authpf.rules**

Defending against brute force attacks

- **table <bruteforce> persist**
- **block quick from ,bruteforce>**
- **pass inet proto tcp from any to $int_if:network prot $tcp_services flags S/SA keep state (max-src-conn 100,**

max src-conn-rate 15/5, overload <bruteforce> flush global)

- **pass quick proto { tcp, udp } from any to any port ssh flags S/SA keep state (max-src-conn 15, max-src-conn-rate 5/3, overload <bruteforce> flush global # tighter for SSH**
- src-conn-rate = connections per X seconds

Tables can be cleaned up to lessen the waste of memory by adding the following to rc.local.

- **/usr/local/sbin/expiretable –v –d –t 24h bruteforce**
- http://expiretable.fnord.se

Defending against spam using spamd.

- tarpit where black listed hosts get stuk
- grey-lists unknowns (mail servers with whom you have never communicated before) using "temporary local error". Real mailhosts will be let through within reasonable time.
- **table <spamd> persist**
- **table <spamd-white> persist file "/var/mail/whitelist.txt"**
- **rdr pass on $ext_if inet proto tcp from <spamd> to { $ext_if, $int_if:network } prot smtp -> 127.0.0.1 port 8025**
- **rdr pass on $ext_if inet proto tcp from !<spamd-white> to { $ext_if, $int_if:network } prot smtp -> 1267.0.0.1 port 8025**

Setting up spamd

- Edit spamd.conf to specify black and white lists
- Enter spamd lines into rc.conf
- Maintenance handled with spamd-setup, fetches lists such as spamhaus, spews (blacklists)
- Your results really depend on the blacklist server you've decided to use. If it sucks, your results will suck :(
- A good one is www.openbsd.org/spamd/traplist.gz