

Secure Network Architectures with OpenBSD

Jason Dixon
DixonGroup Consulting

April 5, 2006

Introduction

- OpenBSD – Secure by Default
- The “Open” Applications
- Link Aggregation, Failover & VLANs
- Packet Filter & Advanced Routing
- IPSec
- IPS/IDS
- “Putting it Together” - Network Design

History of OpenBSD

- A leading secure UNIX-like operating system
- Emphasize code robustness and security
- Open licensing is crucial
- OpenBSD Packet Filter (PF) born out of IPFilter license change



OpenSSH

- Open-source SSH implementation
- Ported to PDAs, commercial appliances, etc.
- Port tunnelling
- SCP/SFTP
- Backups over OpenSSH (tar, dump, rsync, etc)

OpenBGPD

- Free implementations of BGP v4, OSPF v2
- Included with OpenBSD base
- Secure (privilege separation)
- Simple, memory-efficient

OpenNTPD

- Free, simple, secure NTP implementation
- Lean implementation, sufficient for the masses
- Little-to-no configuration required
- Reasonable accuracy

Coming Soon

- OpenOSPF
- OpenCVS
- OpenHTTPD... ?

Packet Filter (PF)

- Packet Normalization
- Filtering
- Network Address Translation
- Advanced Routing & QoS
- Diagnosis & Reporting

Packet Normalization

- Scrub directive normalizes packets, making them “correct” for analysis
- Reassembles fragments
- Drops TCP packets with invalid flag combinations
- Similar to filtering syntax
- Exceptions can be preceded with “no” keyword
- Directives:
 - no-df
 - random-id
 - min-ttl *value*
 - max-mss *value*
 - fragment [reassemble | crop | drop-ovl]
 - reassemble tcp

Filtering

- Macros
- Lists & Tables
- Options
- Tagging
- Stateful Tracking

Filtering (cont'd)

- State tracking options:
 - ♦ *max number*
 - ♦ source-track rule
 - ♦ source-track global
 - ♦ max-src-nodes *number*
 - ♦ max-src-states *number*
 - ♦ max-src-conn *number*
 - ♦ max-src-conn-rate *number / interval*
 - ♦ overload *<table>*
 - ♦ flush [global]

Network Address Translation

- ***nat*** – Standard translation, “many-to-one”
- ***binat*** – Bidirectional mapping, “one-to-one”
- ***rdr*** – Redirect (port-based)
- Translation occurs before filtering
- Targets can be address pools

Quality of Service

- Schedulers
 - ◆ Priority Queueing (priq)
 - Flat structure
 - Service oriented
 - ◆ Class Based Queueing (cbq)
 - Multiple queues or classes
 - Child queues can borrow from parent
- Algorithms
 - ◆ Random Early Detection (red)
 - ◆ Explicit Congestion Notification (ecn)

Quality of Service (cont'd)

- Priq Example:

Root Queue (2Mbps)

Queue A (priority 1)

Queue B (priority 2)

Queue C (priority 3)

etc...

Quality of Service (cont'd)

- CBQ Example:

Root Queue (2Mbps)

UserA (1Mbps)

ssh (50Kbps)

bulk(950Kbps)

User B (1Mbps)

audio (250Kbps)

bulk (750Kbps)

http (100Kbps)

other (650Kbps)

Quality of Service (cont'd)

```
altq on $ext_if priq bandwidth 384Kb queue \  
  { std_out, www_out, ssh_out, dns_out, ack_out }
```

```
queue ack_out priority 7  
queue dns_out priority 6  
queue ssh_out priority 5  
queue www_out priority 4  
queue std_out priority 1
```

```
altq on $int_if priq bandwidth 2.5Mb queue \  
  { std_in, www_in, ssh_in, dns_in }
```

```
queue dns_in priority 7  
queue ssh_in priority 6  
queue www_in priority 5  
queue std_in priority 1
```

Quality of Service (cont'd)

```
block on $ext_if
block on $int_if
pass in on $int_if from ($int_if:network)

pass out on $ext_if inet proto tcp from ($ext_if) to any \
    flags S/SA keep state queue(std_out, ack_out)
pass out on $ext_if inet proto tcp from ($ext_if) to any \
    port { http https } flags S/SA keep state queue www_out
pass out on $ext_if inet proto { udp icmp } from ($ext_if) \
    to any keep state
pass out on $ext_if inet proto { tcp udp } from ($ext_if) \
    to any port domain keep state queue dns_out
pass out on $ext_if inet proto tcp from ($ext_if) to any \
    port ssh flags S/SA keep state queue(std_out, ssh_out)

# continued...
```

Quality of Service (cont'd)

```
pass out on $int_if from any to ($int_if:network)
pass out on $int_if inet proto { tcp udp } from any \
    port domain to ($int_if:network) queue dns_in
pass out on $int_if inet proto tcp from any port ssh \
    to ($int_if:network) queue(std_in, ssh_in)
pass out on $int_if inet proto tcp from any port \
    { http https } to ($int_if:network) keep state queue
    www_in
```

Per-User Filtering (authpf)

- User shell for Gateway Authentication
- Login via OpenSSH, load user anchor ruleset
- User states are removed after the ssh session has closed
- Built-in macros:
 - ♦ \$user_ip
 - ♦ \$user_id

Diagnosis & Reporting

- *pfctl* reveals:
 - ◆ filter, nat, and queue rules
 - ◆ per-rule passed/dropped packet and byte stats
 - ◆ rule numbers, for reference via tcpdump
 - ◆ per label statistics

Diagnosis & Reporting (cont'd)

- *pfctl* also reveals:
 - ◆ table contents
 - ◆ anchor contents
 - ◆ contents of the state table
 - ◆ timeout and memory pool limits
 - ◆ list of OS fingerprints

Diagnosis & Reporting (cont'd)

- *pflogd* logs to the virtual *pflog0* interface
 - ◆ Access to logged packets in real-time with *tcpdump*, including link-level headers
 - ◆ Apply expressions to your pflog logs
 - ◆ See explicit identification of your rules in action

Diagnosis & Reporting (cont'd)

```
# pfctl -si | more
```

```
Status: Enabled for 48 days 11:31:37
```

```
Debug: Urgent
```

```
Interface Stats for vlan1
```

	IPv4	IPv6
Bytes In	84941212208	0
Bytes Out	2434777566	0
Packets In		
Passed	60450379	0
Blocked	855659	0
Packets Out		
Passed	42828680	0
Blocked	691	0

```
State Table
```

	Total	Rate
current entries	8	
searches	205489186	49.1/s
inserts	648818	0.2/s
removals	648810	0.2/s

```
Counters
```

match	7615329	1.8/s
bad-offset	0	0.0/s

Diagnosis & Reporting (cont'd)

```
fragment          14          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         2          0.0/s
proto-cksum       844        0.0/s
state-mismatch    792869      0.2/s
state-insert      5          0.0/s
state-limit       0          0.0/s
src-limit         0          0.0/s
synproxy          0          0.0/s

#
```

Diagnosis & Reporting (cont'd)

```
# pfctl -sr

scrub in all random-id fragment reassemble
block drop in log on vlan1 all
block drop quick from <ssh_trojan> to any
pass out all modulate state
pass in on vlan1 proto tcp from any to (vlan1) port = ssh flags S/SA
    keep state (source-track rule, max-src-conn 10, max-src-conn-rate
    5/30, overload <ssh_trojan> flush global, src.track 30)
pass in on vlan1 proto tcp from any to (vlan1) user = 71 flags S/SA
    keep state
pass in on vlan1 inet proto icmp from any to (vlan1) icmp-type echoreq
    keep state
```

Diagnosis & Reporting (cont'd)

```
# pfctl -vsr

scrub in all random-id fragment reassemble
  [ Evaluations: 65962   Packets: 31204   Bytes: 372028   States: 0 ]
  [ Inserted: uid 0 pid 14263 ]
block drop in log on vlan1 all
  [ Evaluations: 26434   Packets: 415     Bytes: 73384    States: 0 ]
  [ Inserted: uid 0 pid 14263 ]
block drop quick from <ssh_trojan> to any
  [ Evaluations: 26434   Packets: 0       Bytes: 0         States: 0 ]
  [ Inserted: uid 0 pid 14263 ]
pass out all modulate state
  [ Evaluations: 26434   Packets: 5639    Bytes: 843650   States: 0 ]
  [ Inserted: uid 0 pid 14263 ]
pass in on vlan1 proto tcp from any to (vlan1) port = ssh flags S/SA
  keep state (source-track rule, max-src-conn 10, max-src-conn-rate
  5/30, overload <ssh_trojan> flush global, src.track 30)
  [ Evaluations: 26434   Packets: 868     Bytes: 121207   States: 1 ]
  [ Inserted: uid 0 pid 14263 ]
```

Diagnosis & Reporting (cont'd)

```
# pfctl -vvvsr

@0 scrub in all random-id fragment reassemble
  [ Evaluations: 66478   Packets: 31463   Bytes: 378412   States: 0 ]
  [ Inserted: uid 0 pid 14263 ]
@1 block drop in log on vlan1 all
  [ Evaluations: 26536   Packets: 419     Bytes: 73600    States: 0 ]
  [ Inserted: uid 0 pid 14263 ]
@2 block drop quick from <ssh_trojan:0> to any
  [ Evaluations: 26536   Packets: 0       Bytes: 0         States: 0 ]
  [ Inserted: uid 0 pid 14263 ]
@3 pass out all modulate state
  [ Evaluations: 26536   Packets: 5650    Bytes: 844486   States: 3 ]
  [ Inserted: uid 0 pid 14263 ]
@4 pass in on vlan1 proto tcp from any to (vlan1:2) port = ssh flags
  S/SA keep state (source-track rule, max-src-conn 10, max-src-conn-
  rate 5/30, overload <ssh_trojan> flush global, src.track 30)
  [ Evaluations: 26536   Packets: 1018    Bytes: 136575   States: 1 ]
  [ Inserted: uid 0 pid 14263 ]
```

Diagnosis & Reporting (cont'd)

```
# pfctl -vsq | grep -A1 out
```

```
queue std_out      priq( default )  
[ pkts:    26615  bytes:   4068428  dropped pkts:    58  bytes:    5423 ]  
queue vpn_out      priority 2  
[ pkts:   120513  bytes: 105441372  dropped pkts:    13  bytes:   15606 ]  
queue mail_out     priority 3  
[ pkts:    18985  bytes:   1748608  dropped pkts:     0  bytes:     0 ]  
queue http_out     priority 4  
[ pkts:    47143  bytes:   5186482  dropped pkts:     0  bytes:     0 ]  
queue ssh_out      priority 5  
[ pkts:     6725  bytes:    492474  dropped pkts:     0  bytes:     0 ]  
queue dns_out      priority 6  
[ pkts:     2149  bytes:    180428  dropped pkts:     0  bytes:     0 ]  
queue tcp_ack_out  priority 7  
[ pkts:     491  bytes:    31614  dropped pkts:     0  bytes:     0 ]
```

Diagnosis & Reporting (cont'd)

```
# grep label /etc/pf.conf

pass in on $ext_if inet proto tcp from any to $dmz port $tcp_svcs
    flags S/SA keep state label "dmz-IN:$dstport"
pass in on $ext_if inet proto icmp all icmp-type echoreq keep state
    label "all-IN:echoreq"

# pfctl -vsq | grep -A1 out

dmz-IN:20 22 0 0
dmz-IN:21 24 0 0
dmz-IN:22 20 351 46918
dmz-IN:25 20 21 1234
dmz-IN:80 20 186 93678
dmz-IN:443 20 0 0
dmz-IN:465 20 0 0
dmz-IN:993 20 0 0
dmz-IN:995 20 115 12855
all-IN:echoreq 31 46 3864
```

Diagnosis & Reporting (cont'd)

```
# pfctl -ss

self icmp 144.206.140.239:50657 <- 71.238.24.24:50657 0:0
self icmp 144.206.140.239:50658 <- 71.238.24.24:50658 0:0
self tcp 10.200.200.1:22 <- 10.200.200.2:45682 ESTABLISHED:ESTABLISHED
self tcp 10.200.200.1:22 <- 10.200.200.2:45724 FIN_WAIT_2:FIN_WAIT_2
self tcp 144.206.140.239:45716 -> 66.151.150.22:2703 TIME_WAIT:TIME_WAIT
self tcp 144.206.140.239:22 <- 71.238.24.24:50609 ESTABLISHED:ESTABLISHED
self tcp 144.206.140.239:995 <- 71.238.24.24:50653 TIME_WAIT:TIME_WAIT
self tcp 144.206.140.239:45722 -> 68.142.226.32:80 FIN_WAIT_2:FIN_WAIT_2
self tcp 144.206.140.239:45737 -> 82.165.144.45:80 FIN_WAIT_2:FIN_WAIT_2
self tcp 144.206.140.239:995 <- 72.255.47.155:1634 TIME_WAIT:TIME_WAIT
self tcp 144.206.140.239:995 <- 72.255.47.155:1648 TIME_WAIT:TIME_WAIT
self tcp 144.206.140.239:25 <- 61.37.207.162:2988 FIN_WAIT_2:FIN_WAIT_2
self tcp 144.206.140.239:45736 -> 208.254.3.166:80 FIN_WAIT_2:FIN_WAIT_2
self udp 144.206.140.239:56099 -> 144.202.253.2:53 MULTIPLE:MULTIPLE
self udp 144.206.140.239:56108 -> 144.202.253.2:53 MULTIPLE:MULTIPLE
```

Diagnosis & Reporting (cont'd)

```
# pfctl -st

tcp.first          120s
tcp.opening        30s
tcp.established    86400s
tcp.closing        900s
tcp.finwait        45s
tcp.closed         90s
tcp.tsdiff         30s
udp.first          60s
udp.single         30s
udp.multiple       60s
icmp.first         20s
icmp.error         10s
other.first        60s
other.single       30s
other.multiple     60s
frag              30s
interval          10s
adaptive.start     0 states
adaptive.end       0 states
src.track          0s
```

Diagnosis & Reporting (cont'd)

```
# tcpdump -nettr /var/log/pflog port 1025
Feb 28 13:37:03.508510 rule 0/(match) block in on vlan0:
    222.174.34.149.38701 > 10.90.90.45.1025:  udp 479 (DF)
Feb 28 13:37:03.532619 rule 0/(match) block in on vlan0:
    222.174.34.149.38701 > 10.90.90.50.1025:  udp 479 (DF)
Feb 28 13:37:03.717936 rule 0/(match) block in on vlan0:
    222.174.34.149.38701 > 10.0.0.207.1025:  udp 479 (DF)
Feb 28 13:37:03.833415 rule 0/(match) block in on vlan0:
    222.174.34.149.38701 > 198.207.179.10.1025:  udp 479 (DF)
```

Bridges

- Bridge L2 traffic between physical segments
- A physical interface can route AND bridge

```
# ifconfig em0 up
# ifconfig em1 up
# brconfig bridge0 add em0 add em1

# cat /etc/hostname.em*
up
up

# cat /etc/bridgename.bridge0
add em0
add em1
blocknonip em0
rule pass in on em1 src 1:2:3:4:5:6 tag boss
```

Trunks

- Link aggregation and failover

```
# ifconfig fxp0 up
# ifconfig fxp1 up
# ifconfig trunk0 trunkport fxp0 trunkport fxp1 up
# ifconfig vlan0 10.0.0.2 netmask 255.255.255.0 \
    vlan 100 vlandev trunk0

# cat /etc/hostname.*
up
up
trunkport fxp0 trunkport fxp1 up
inet 10.0.0.2 255.255.255.0 10.0.0.255 \
    vlan 100 vlandev trunk0 up
```

Trunks (cont'd)

```
# ifconfig fxp

fxp0: flags=8943<UP,BROADCAST,RUNNING,PROMISC,SIMPLEX,MULTICAST> mtu
1500
    lladdr 00:02:b3:0a:ce:04
    trunk: trunkdev trunk0
    media: Ethernet autoselect (100baseTX full-duplex)
    status: active
    inet6 fe80::202:b3ff:fe0a:ce04%fxp0 prefixlen 64 scopeid 0x1

fxp1: flags=8943<UP,BROADCAST,RUNNING,PROMISC,SIMPLEX,MULTICAST> mtu
1500
    lladdr 00:02:b3:16:a9:57
    trunk: trunkdev trunk0
    media: Ethernet autoselect (100baseTX full-duplex)
    status: active
    inet6 fe80::202:b3ff:fe16:a957%fxp1 prefixlen 64 scopeid 0x2
```

Trunks (cont'd)

```
# ifconfig trunk

trunk0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    lladdr 00:02:b3:0a:ce:04
    trunk: trunkproto roundrobin
           trunkport fxp1
           trunkport fxp0 master
    groups: trunk
    media: Ethernet autoselect
    status: active

# ifconfig vlan

vlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1496
    lladdr 00:02:b3:0a:ce:04
    vlan: 100 parent interface: trunk0
    groups: vlan egress
    inet6 fe80::202:b3ff:fe0a:ce04%vlan0 prefixlen 64 scopeid 0x9
    inet 10.0.0.2 netmask 0xffffffff broadcast 10.0.0.255
```

VLANs

- IEEE 802.1Q virtual interface
- Useful for filtering on many networks with one dedicated NIC
- Can serve as a parent interface for other virtual interfaces (carp)

```
# ifconfig vlan0 10.0.0.2 netmask 255.255.255.0 \  
    vlan 100 vlandev em0  
  
# cat /etc/hostname.vlan0  
inet 10.0.0.2 255.255.255.0 10.0.0.255 vlan 100 vlandev em0  
inet alias 10.0.0.3  
inet alias 10.0.0.4  
inet alias 10.0.0.5
```

CARP

- Common Address Redundancy Protocol
- Free alternative to VRRP
- Virtual MAC based on vhid
- Failover, Preemption, Load Balancing
- States – MASTER, BACKUP, INIT

PFSYNC

- Synchronizes state tables
- Multicast PFSYNC announcements
- Can be routed to peer with *syncpeer*

IPSec

- ISAKMP (isakmpd)
- ipsecctl
- sasyncd

Applications

- *spamd* – spam tarpit, supports greylisting
- *httpd* – patched Apache-1.3, chrooted by default
- *named* – patched Bind-9, chrooted by default
- *ftp-proxy* – supports active/passive modes in front of either client or server

IPS/IDS

- /etc/daily
- /etc/security
- Snort
- SnortSam
- snort2c

References

- OpenBSD -- <http://www.openbsd.org/>
- PF -- <http://www.benedrine.cx/pf.html>
- OpenBSD FAQ -- <http://www.openbsd.org/faq/index.html>
- PF User's Guide -- <http://www.openbsd.org/faq/pf/index.html>