

Building a Home Firewall/Router Using OpenBSD-Sparc



Joshua Malone
(jmalone@ubergeeks.com)

Sun Hardware



- Cheap! (ebay, surplus auctions, giveaway)
- Reliable, enterprise-grade hardware
- OS options
 - SunOS / Solaris (ugh)
 - OpenBSD (**best** OS for sparc HW – very well ported)
 - NetBSD (sparc32 port)
 - Linux (SuSE, older Red-Hat, Debian)

Sun Hardware cont'

- Expansion
 - SCSI (almost every Sun box has it on-board)
 - S-bus (network cards, extra SCSI busses, framebuffer)
 - Framebuffer is like video card for Sun – but not necessary; you can use serial console and run headless!
 - M-bus – add CPUs! (some machines)
 - M-bus CPU modules available on ebay and other places
 - Only NetBSD-2 and Linux can utilize multiple CPUs

Sun Hardware cont'

Limitations:

- S-bus is 25 Mhz
 - Even so, it can handle a 768Kbit/sec DSL line
- No way to get USB
 - Need a cable/DSL modem with ethernet
- Slow bootup (compared to a so-ho router appliance)
 - Need to disable a bunch of un-needed startup scripts

Need wireless? Just add an access point.

Sun Sparcstation LX

- 50 MHz Micro-sparc CPU (soldered on board)
- Sun-4/m type (4/30)
 - introduced 11/92, end-of-lifed 3/94
- RAM: up to 96MB (I'm using 48)
- 1 internal SCSI HD (I have a 1GB Seagate)
- On-board 10 base-T ethernet
- On-board Cgsix framebuffer
 - But we'll be going headless!

Typographic conventions

- <italic in angle>* - Parameter – replace with your specific value
- {values|in|braces} - Choices for a value – choose one
- [square brackets] - Optional item
- Fixed type - Command or line for a file – literal

OpenBSD

www.openbsd.org

- Free, BSD-style operating system focusing on security by default
 - Heavily crypto-oriented
 - Basic install is < 250MB, including:
 - pf
 - dhcpcd
 - bind
 - ssh
- Perfect for older,
under-powered hardware!

Installing OpenBSD-sparc

- Need serial, null-modem cable
 - 2 types of serial ports: DB-25 and mini DIN-8
 - DB25 is actually 2 serial ports in one
 - Normal cable only connects first one
 - Mac serial cable should work with mini DIN-8
- Single net-install floppy
 - OpenBoot: 'boot floppy'

Configuring OpenBSD

- `/etc/hostname.<ifname>`
 - Per-interface network configuration
 - Format: `inet <IP> <mask> [<bcast>] [<options>]`
 - For DHCP: `'dhcp NONE NONE NONE'`
- `/etc/mygate`
 - Default gateway info – just list the IP
- `/etc/myname`
 - System hostname

PPP over Ethernet

- /etc/ppp.conf

default:

```
set log Phase Chat IPCP CCP tun
  command
set redial 15 0
set reconnect 15 10000
```

pppoe:

```
set device "!/usr/sbin/pppoe -i le1"
disable acfcomp protocomp
deny acfcomp
set mtu max 1492
set speed sync
enable lqr
set lqrperiod 5
set cd 5
set dial
set login
set timeout 0
set authname <username>
set authkey <password>
add! default HISADDR
enable dns
enable mssfixup
```

External
Interface

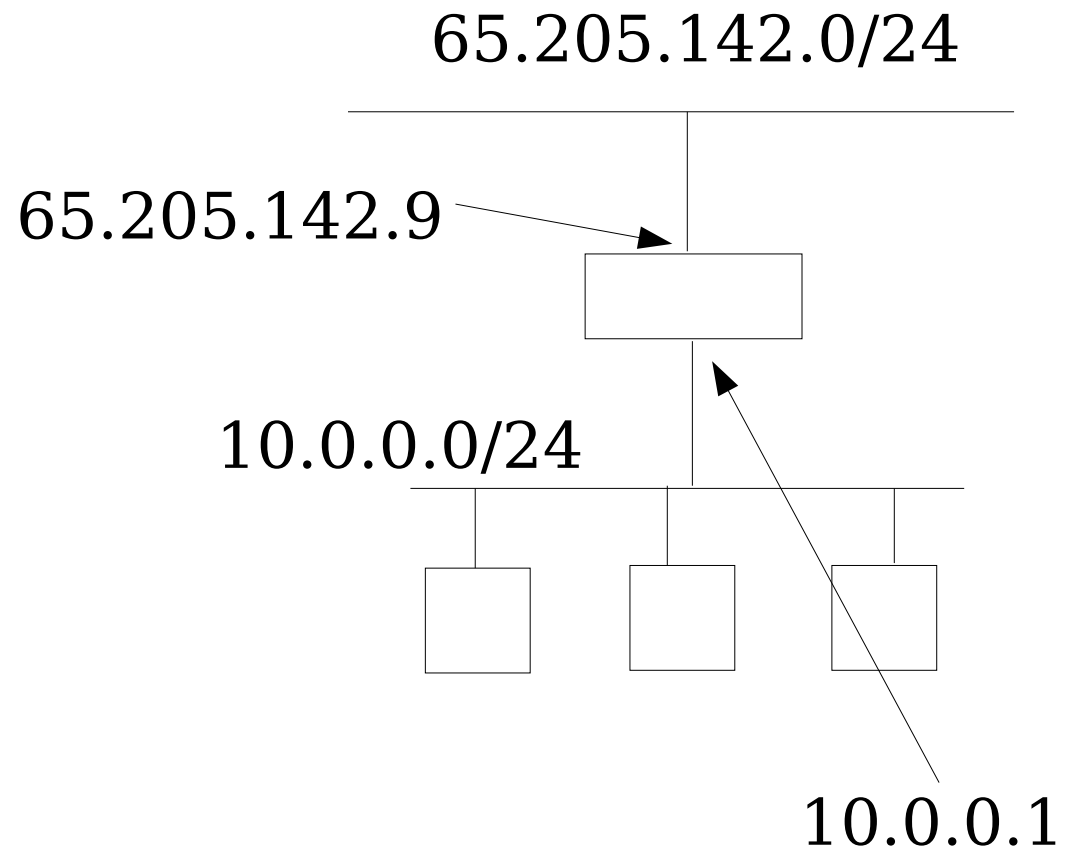


Compensate for PPP
overhead



NAT Basics

- Network Address Translation
- Firewall translates the internal private IPs into an assigned public one
- Also called IP 'Masquerading'
- “Hides” the internal IP space and the internal hosts.



PF Basics

- `/etc/rc.conf`
 - `pf=YES` (enables pf at boot time)
 - Test first - don't lock yourself out!
- `/etc/pf.rules`
 - All rules for filtering and nat
- `/etc/sysctl.conf`
 - `net.inet.ip.forwarding=1`

PF Basics

- Turn on/off
 - Enable: 'pfctl -e' Disable: 'pfctl -d'
- Load/flush rules
 - Load rules: 'pfctl -f <filename>' (*Same as IPF*)
 - Flush filter rules: 'pfctl -F rules'
 - Flush nat rules: 'pfctl -F nat'
 - Flush state table: 'pfctl -F state'
 - Show loaded ruleset: 'pfctl -s rules'
- Use 'pfctl' for both filter *and* nat

PF Configuration

- Last-match wins!
- Basic rules
 - {block|pass} [{in|out}] [quick] [on <if>] [inet] [proto {tcp|udp}] from <source> to <dest> [keep state]
 - 'quick' – apply this rule and stop checking the rule list
 - inet: must be specified for some proto's (icmp)
 - source/dest can be IP or network in CIDR notation
 - source/dest can have 'port #' after them
 - 'keep state' – adds connection to the state table and expects a reply to the conversation
 - works with TCP and UDP

PF Configuration

- Blocking options:
 - block [*<policy>*]
 - Policy types:
 - drop (the default): Do nothing, just ignore the packet
 - return: Return a TCP RST or ICMP UNREACH depending on type
 - return-rst: Return TCP RST – only use for proto tcp
 - return-icmp: Return ICMP UNREACH – only use for proto udp

```
block return-rst in quick proto tcp from  
any to any port 23
```

Block telnet

PF Configuration

- NAT

- nat on *<ext if>* from *<internal net>* to any -> *<ADDR>*

- ADDR is address to nat to; if you have dynamic IP, use '*<if>*'

- nat on le1 from 10.0.0.0/24 to any -> (le1)

- Port Forwarding

- rdr on *<input interface>* proto {tcp|udp} from any to *<IP>*
port *<portnum>* -> *<internal address>*

- rdr on le1 proto tcp from any to any port 80
-> 10.0.0.45

Advanced PF Syntax

- **Macros!**

- `RFC1918 = "{ 192.168.0.0/16, 172.16.0.0/12, 10.0.0.0/8 }"`
- `block in quick on $EXT from $RFC1918 to any`

- **Packet scrubbing**

- De-fragment packets before passing them; protect machines from fragment attacks (DOS, etc.)
- `scrub {in|out}`

- **Antispoof**

- Drop packets that arrive on the wrong interface
- `antispoof for <interface>`

Minimal Firewall Configuration

```
RFC1918 = "{ 192.168.0.0/16, 172.16.0.0/12, 10.0.0.0/8 }"  
lannet = "{ 10.0.8.0/24 }"  
INT = "le0"  
EXT = "le1"  
  
# Nat to the WAN  
nat on $EXT from $lannet to any -> ($EXT)  
  
# Default to blocking - catch-all rule  
block log all  
  
pass quick on lo0 all  
pass in quick on $INT from $lannet to any  
pass out quick on $INT from any to $lannet  
  
block in quick on $EXT from $RFC1918 to any  
  
# Accept pings  
pass in on $EXT inet proto icmp all icmp-type echoreq keep state  
  
pass out on $EXT proto tcp keep state flags S/SA  
pass out on $EXT proto { udp, icmp } keep state
```

Debugging Rulesets

No equivalent to 'ipmon' for pf

Use 'tcpdump' to view pf logs:

- `tcpdump -n -e -ttt -i pflog0`
 - n – don't resolve ips to hostnames, etc.
 - e – print link-level header info
 - ttt – print human-readable timestamp

Debugging Rulesets

Example: Telnet blocked

```
$> tcpdump -n -e -ttt -i pflog0
tcpdump: WARNING: pflog0: no IPv4 address assigned
tcpdump: listening on pflog0
Jul 26 15:33:59.568436 rule 8/0(match): block in on le0:
  10.0.8.100.32778 > 10.0.8.1.23: S 701220195:701220195(0)
  win 5840 <mss 1460,sackOK,timestamp[|tcp]> (DF) [tos 0x10]
Jul 26 15:34:02.565304 rule 8/0(match): block in on le0:
  10.0.8.100.32778 > 10.0.8.1.23: S 701220195:701220195(0)
  win 5840 <mss 1460,sackOK,timestamp[|tcp]> (DF) [tos 0x10]
^C
2 packets received by filter
0 packets dropped by kernel
```

Debugging Rulesets

Example: Telnet blocked

```
$> pfctl -s rules
0 block drop in on ! le0 inet from 10.0.8.0/24 to any
1 block drop in inet from 10.0.8.1 to any
2 block drop in on le0 inet6 from fe80::a00:20ff:fe1d:d663 to any
3 block drop log all
4 block drop in log all
5 block return-rst in log on le1 proto tcp all
6 pass quick on lo0 all
7 pass in on le1 inet proto icmp all icmp-type echoreq keep state
8 block drop in log quick on le0 proto tcp from any to any port = telnet
pass in quick on le0 inet from 10.0.8.0/24 to any
pass out quick on le0 inet from any to 10.0.8.0/24
block drop in log quick on le1 inet from 172.16.0.0/12 to any
block drop in log quick on le1 inet from 10.0.0.0/8 to any
pass out on le1 proto tcp all flags S/SA keep state
pass out on le1 proto udp all keep state
pass out on le1 proto icmp all keep state
```

Adding DHCP

Allows computers to auto-configure network

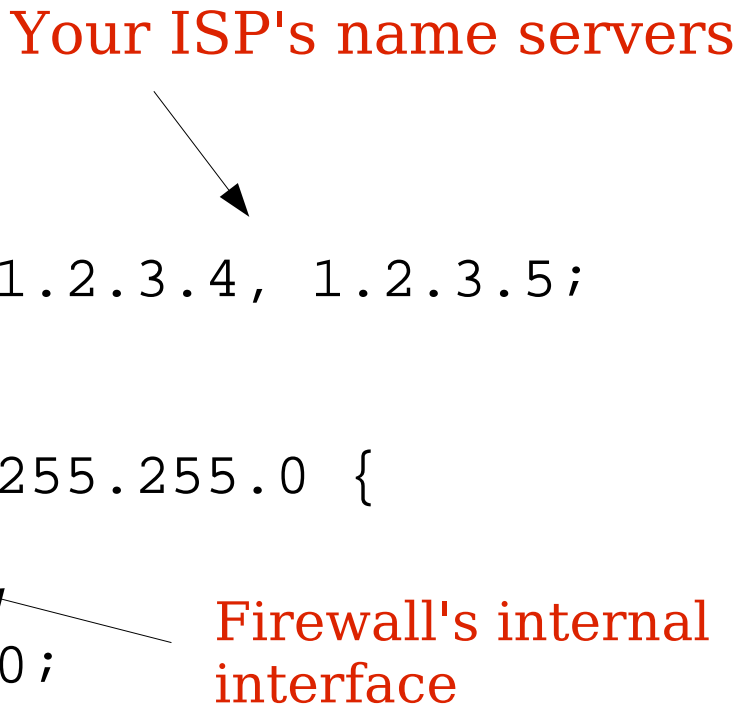
- /etc/dhcpd.interfaces
 - just list the name of the **INTERNAL** interface (le0, rl1, etc.)
- /etc/dhcpd.conf
- /etc/rc.conf
 - `dhcpd_flags=' -q '`

Minimal DHCPd Configuration

```
shared-network LOCAL-NET {  
    option domain-name "foobar";  
    option domain-name-servers 1.2.3.4, 1.2.3.5;  
  
    subnet 10.0.0.0 netmask 255.255.255.0 {  
        option routers 10.0.0.1;  
        range 10.0.0.30 10.0.0.250;  
    }  
}
```

Your ISP's name servers

Firewall's internal interface



- Good idea not to use a resolvable domain name for private network

Allowing DHCP through a filter

- Uses UDP, ports 67 & 68
 - Initial traffic will be between 0.0.0.0 and 255.255.255.255!
 - For receiving DHCP requests (DHCP server)

```
pass in quick on le0 proto udp from any port 67 to any port 68  
keep state
```

- For making DHCP requests (DHCP client – dynamic IP)

```
pass out quick on le1 proto udp from 0.0.0.0 to 255.255.255.255  
port 67 keep state
```

Adding Internal DNS

OpenBSD 3.5 ships with BIND-9.2.3 stock

- `/var/named`
 - `etc/named.conf`
 - (`etc/named-simple.conf`)
- Can also run in caching-only mode
 - Speed common lookups
 - Provide internal nameserver for dhcpd to assign

For information about configuring BIND, see www.isc.org/index.pl?/sw/bind/

Dynamic DNS

- Allows you to have a forward DNS record even if you don't have a static IP!
- Providers:
 - dyndns.org, no-ip.com, 2mydns.com
 - Good list at www.technopagan.org/dynamic/
- Client on the firewall sends DDNS update to the provider to register your new IP every time it changes.

DDNS Clients

- **ddclient** (linux.cudeso.be/linuxdoc/ddclient.php)
 - perl
- **ipcheck** (ipcheck.sourceforge.net)
 - python
- **Check OpenBSD ports collection for more**
 - www.openbsd.org/ports.html

Resources

- www.openbsd.org/sparc.html
- real.ath.cx/BSDinstall.html
- www.obsolyte.com (Sun hardware info)
- www.inebriated.demon.nl/pf-howto/
- www.drones.com/obsd-fw.html
- www.muine.org/~hoang/openpf.html
- www.isc.org/index.pl?/sw/dhcp/

Sun, Sun Microsystems and the Sun logo are trademarks of Sun Microsystems, Inc.
The OpenBSD logo is a trademark of the OpenBSD project.