Global Information Assurance Certification Paper

Setting up a Firewall with IPF, IPNAT and OpenBSD Version 1

Jason van Brecht

26th June 2001

Introduction

We have come a very long way since the early days of the Internet, when users on the Net could trust each other and little security was needed. However, today inexpensive high speed bandwidth is readily available, computers are cheap, and people are telecommuting from home, and that data needs to be secured from prying eyes. I will discuss in detail how to setup a cost effective firewall to protect your network using OpenBSD, IPF and IPNAT. Installation of the operating system will not be covered, that information is available http://www.openbsd.org/faq/faq4.html.

Overview

OpenBSD is an open source secure operating system that is a BSD 4.4 like operating system that runs on multiple hardware platforms

OpenBSD will run on the following platforms:

- Intel based systems (i386),
- Amiga m68k-based models (amiga)
- Motorola MVME147/16x/17x 68K VME cards (mvme68k)
- Hewlett-Packard HP300/HP400 machines (hp300)
- Support for Apple based PowerPC systems (powerpc)
- SPARC Platform by Sun Microsystems (sparc)
- Sun's 68020 based Sun3 models (sun3)
- DEC's VAX computers. (vax)

"Four years without a remote hole in the default install!"

IPF is a powerful packet filter that comes as part of the distribution, and is used to insert or remove simple to very complex packet filtering rules that can protect a whole network. Depending on the ruleset you create, IPF can be used to block or permit inbound or outbound packets depending on their protocol, protocol options, can track fragments and apply a rule to all fragments of that particuar packet, to or from any network.

IPNat is the utility that controls the kernels network address translation tables. This allows people to use non routable internal IP addresses (RFC1918) address for internal networks that do not require real IP's, or the user has more then one machine and only has one IP. IPNAT and IPF will work with any type of interface from Ethernet to dialup.

Basic Setup and Installation

OpenBSD can be installed numerous ways, from FTP install to CDROM install. Since you are setting up a firewall, you have network connectivity, so we will be doing a network install via FTP. You will need one of these three files, floppy29.fs, floppyB29.fs or floppyC29.fs. In most cases you will only need floppy29.fs unless you are running some obscure hardware. Explanation of what hardware is supported on the boot floppies is here under heading 4.1.3

To create the boot disks:
Unix "dd if=/path/to/floppy29.fs of=/dev/fd0" (replace fd0 with your floppy drive device)
Windows 95 users can use this tool, rawrite, NT users can use fdimage to create the disks.

Once your boot disk is created, place it in the disk drive, make sure the machine is setup to boot from the floppy drive and reboot the system.

The basic requirement is any system listed above with 2 communication devices (modem, ethernet etc). A video card and monitor are needed for the install, however once the system is setup they can be removed.

Setting up OpenBSD
Installation of OpenBSD itself will not be covered in this paper, that information can be found at http://www.openbsd.org/faq/faq4.html, the OpenBSD installation guide.

Once the basic installation has been completed, make sure you have the latest stable source and ports. The following command will pull down the latest source and ports
The cvsup binary can be obtained here
cvsup –g –L 2 stable-ports.update
Where **stable-ports.update** is the file you created (example at the bottom of this document)

Setting up the kernel
Next we make sure that a few options are compiled into the kernel. Change directory to */usr/src/sys/arch/i386/conf*. Your will see a few files there, namely GENERIC. Copy GENERIC to a new name, YOUR_HOSTNAME is always a good choice, and edit the file you just created. There are many options, keep and remove the ones for your hardware, however add the following to your kernel if they are not already there. (side note, sometimes at the beginning of the generic kernel you used there is a line that includes another config file, like *include "../../../conf/GENERIC"*, the options listed above are often already in the default kernel, but in case they are not you will need to add them). If the options listed above are already compiled in, then you can skip this section

options IPFIREWALL_VERBOSE
options IPFILTER
options IPFILTER_LOG
options TCP_DROP_SYNFIN
options TCP_RESTRICT_RST
options ICMP_BANDLIM

.
IPFIREWALL_VERBOSE enables the firewall to print output, IPFILTER enables ipf, IPFILTER_LOG enables ipf logging, TCP_DROP_SYNFIN drops all packets with the syn and fin set, TCP_RESTRICT_RST drops packets with rst options set, and lastly

ICMP_BANDLIM limits the amount of icmp coming into the machine. These options are mostly to lower the chances of being "denial or service" attacked.
Compile the kernel, install it and reboot.

Next we disable everything we do not need, the default install enables a lot of services, we are going to disable those. Most of what runs on bootup, runs from the rc.conf file in /etc. Search for these lines and change their option to NO if they are set to YES.
inetd=YES
Inetd is almost always enabled by default, change it to "***inetd=NO***"
Identd runs out of inetd, there really is no reason to run it on a firewall, but if you must, look for **identd_flags** and make it ="**-B –u nobody –elo**"
portmap=YES
As with inetd, portmap is also almost always enabled, change it to ***portmap=NO***

Next we need to enable the ipfilter and ipnat abilities, again in /etc/rc.conf , change the NO to a YES for the following that default to NO. This tells the system to use the conf files /etc/ipnat.rules and /etc/ipf.rules on bootup (if you left ipfilter and ipnat=no, then you would have to manually enable ipnat and ipf.
ipfilter=YES
ipnat=YES
Next, edit the file ***/etc/sysctl.conf*** and change ***net.inet.ip.forwarding=0*** to ***net.inet.ip.forwarding=1***

Once the files have been changed, reboot the system. During the boot sequence you should see the following lines, if you do not, check the above instructions to see if anything was missed.
configuring IP filter
<misc program startup messages>
configuring NAT
0 entries flushed from NAT tables
0 entries flushed from NAT list
By default on the initial system install, a rule of all is permitted in and out, which will be applied at boot time. Always make sure when modifying firewall rules you have a method or a person to reset the system or change the rules back in case you lock yourself out. It is always best to make changes to firewall rulesets locally on the firewall, and preferably at the most inconvenient time in case you end up dropping all traffic to your network.

Setting up Network Address Translation (NAT) Rules
From here on out, my outside interface will be using xl0 as my outside interface, and xl1 as my inside interface (2 3com nic's)
The syntax for the file ***/etc/ipnat.rules*** is as follows:
map interface internal/mask -> external/mask options
map tells ipnat how an address range should be translated.
interface is the external interface, xl0

internal/mask is your internal network addressing scheme, 10.0.0.0/24 will map 10.0.0.0 through 10.0.0.255
external/mask is your external addressing. Using interface/32 will map your internal addresses to whatever address your interface is assigned.
Options, both the internal and external can be IP addresses, however a problem may arise when you attempt to map say a full class A of 16 million addresses through a single class C of 254 addresses, this is where the options come into play, by remapping using ports instead, using the **portmap** option. You can use tcp, udp or tcp/udp, the port range syntax is portnumber:portnumber. In our example we are mapping 254 ip's to a single IP. Below would map 10.0.0.* to whatever ip xl0 has (this can be assigned or given using dhcp)

The basic ruleset for ***/etc/ipnat.rules***

map xl0 10.0.0.0/24 -> xl0/32 portmap tcp/udp 10000:20000
map xl0 10.0.0.0/24 -> xl0/32

Further more, if you wanted to run any type of server, like a web server or mail server inside your network, ipnat can port forward. Say for example 10.0.0.50 is your web server, it can be accessible from the outside with the following line in your ipnat.rules.

rdr xl0 0.0.0.0/32 port 80 -> 10.0.0.50 port 80

Now, if xl0 is a real ip, any traffic to that ip on port 80 would be redirected to the web server at 10.0.0.50. This can be done for any service, and redirected to any machine. The next line is the built in ftp proxy. This will allow you to ftp from the inside of your network without having to use passive mode.

map xl0 10.0.0.0/24 -> proxy port ftp ftp/tcp

So, in the end you should have a file that looks like this

map xl0 10.0.0.0/24 -> xl0/32 proxy port ftp ftp/tcp
map xl0 10.0.0.0/24 -> xl0/32 portmap tcp/udp 1025:65000
map xl0 10.0.0.0/24 -> xl0/32
rdr xl0 0.0.0.0/0 port 80 -> 10.0.0.50 port 80
rdr xl0 0.0.0.0/0 port 22 -> 10.0.0.49 port 22

This will map any ip from 10.0.0.* to the ip address or xl0, will forward any requests on port 80 to machine .50 and any requests on port 22 will be forwarded to machine .49

Setting up IPF (IP Filter)
The rules are read from top to bottom, even after a packet has been matched to a rule, the rest of the rules are still processed unless the keyword *quick* is used.
Take the following example:

block in all
pass in all

The above ruleset will permit all traffic through, a packet will be matched to the first rule **block in all**, then the next rule will be applied to the packet, that being **pass in all**, which would permit the traffic through as rules are read from top to bottom, the rules lower down taking preference. When the *quick* keyword is used, packets are matched to the rule and then the next packet is processed, not the next rule. This will reduce system overhead when you have a very large ruleset, so rather then applying a few hundred lines (if your ruleset is that big) of a ruleset to every packet, we use the keyword of *quick*. When a packet is matched to a rule with the keyword *quick*, the rest of the rules are ignored and the system moves on to the next packet. Take the following example where xl0 is your outside interface and v.w.x.y/z is your ip subnet

block in rules
block in quick on xl0 from 192.168.0.0/16 to any (RFC1918 private addresses)
block in quick on xl0 from 10.0.0.0/8 to any (RFC1918 private addresses)
block in quick on xl0 from 172.16.0.0/12 to any (RFC1918 private addresses)
block in quick on xl0 from 127.0.0.0/8 to any (Loopback address space)
block in quick on xl0 from 0.0.0.0/8 to any
block in quick on xl0 from 192.0.2.0/24 to any (Reserved IP Block)
block in quick on xl0 from 204.152.64.0/23 to any(Reserved by Sun for private clusters)
block in quick on xl0 from 224.0.0.0/3 to any (RFC1166 address space)
block in quick on xl0 from v.w.x.y/z to any (Your own address space)
pass in all

The addresses listed above are address spaces that you should never see on your outside interface, so any packets that get matched to those rules are going to be spoofed. But this is only half a firewall, that will stop the traffic from getting in, depending on how many users you have, and how trusting you are, you may want to drop spoofed traffic leaving your network as well. Simply by rearranging the ruleset a little, we can accomplish this, since the traffic that should not be let in, should also not be let out.

block out rules
block out quick on xl0 from any to 192.168.0.0/16

For each of the original lines EXCEPT your own subnet, replace the **in** with **out** and swap the **subnet to any** to **any to subnet**.
This is will be a basic firewall that will drop traffic that should not be routed over the internet, and permit all other traffic through in both directions. So your end result will be something like this

block out rules
pass out quick on xl0 from v.w.x.y/z to any
block in rules
pass in all

This type of firewall is not very useful, since it will allow any type of traffic with legitimate ip's both in and out of the firewall.

The next set of rules will include the keyword *keep state*, this keyword keeps track of outgoing sessions, and has the ability to keep track of individual sessions using tcp/udp/icmp. Once a connections status has entered the state tables, the packets from the session are permitted through the firewall regardless of what your ruleset says. Keeping the state permits you to initiate an outgoing connection, and forget about the return traffic, otherwise a separate rule would have had to be created for returning traffic.

block in quick on xl0 all
pass out quick on xl0 proto tcp from v.w.x.y/z to any keep state
pass out quick on xl0 proto udp from v.w.x.y/z to any keep state
pass out quick on xl0 proto icmp from v.w.x.y/z to any keep state

These rules will drop all incoming traffic, but permit users inside your network complete and unrestricted access to the outside networks. This however will not allow you to run any services on the inside of your network. So what we need to do is create paths through the firewall to specific machines.

pass in quick on xl0 proto tcp from any to v.w.x.y/z port = 80 keep state
pass in quick on xl0 proto tcp from any to v.w.x.y/z port = 25 keep state
block in log quick on xl0 all
pass out quick on xl0 proto tcp from v.w.x.y/z to any keep state
pass out quick on xl0 proto udp from v.w.x.y/z to any keep state
pass out quick on xl0 proto icmp from v.w.x.y/z to any keep state

The above ruleset will permit incoming traffic to machines v.w.x.y/z which would be those machines running your services, in this case http and smtp traffic would be permitted in but all other traffic would be dropped, and logged to the ipfilter logging device /dev/ipl, which can be read using ipmon –s (and –D for ipmon top fork as a daemon). However, since ipf 3.3, the ability to log directly to syslog without requiring third party programs was added.

block in log level local0.info quick on xl0 all

This will send all ipf logs to where ever you are logging the local0 facility, and info level. The line in your */etc/syslog.conf* would look something like this. (for more info see syslog help and man pages on what facility and level are)

local7.auth /var/log/ipflog

Make sure the file ipflog exists before restarting the syslog daemon. When viewing the rules, you will see a line similair to this one

Jun 29 12:13:56 gw ipmon[20546]: 12:13:55.806218 tun0 @0:14 b 66.22.112.13,1222 -> 208.59.233.179,23 PR tcp len 20 44 -S IN

Note, the rule that was applied to this packet is stated in this part of the line, **tun0 @0:14 b**, states that a packet from machine 66.x.x.13 was block on interface tun0 at line 14. So when fine tuning your rules, check your log files to see what is being dropped when there is something that should be permitted. (when they list a rule line, do not include lines that start with #, these are comments and are not counted)

(ipmon which is the tool that reads /dev/ipl logs to local0.info by default which may or may not already be in your syslog.conf)

So lets put all the above rules into a file, so that we can have a usefull and simple ruleset.

block in quick on xl0 from 192.168.0.0/16 to any
block in quick on xl0 from 10.0.0.0/8 to any
block in quick on xl0 from 172.16.0.0/12 to any
block in quick on xl0 from 127.0.0.0/8 to any
block in quick on xl0 from 0.0.0.0/8 to any
block in quick on xl0 from 192.0.2.0/24 to any
block in quick on xl0 from 204.152.64.0/23 to any
block in quick on xl0 from 224.0.0.0/3 to any
block in quick on xl0 from v.w.x.y/z to any
pass in quick on xl0 proto tcp from any to v.w.x.y/z port = 25 keep state
pass in quick on xl0 proto tcp from any to v.w.x.y/z port = 80 keep state
pass in quick on xl0 proto tcp from any to v.w.x.y/z port = 53 flags S keep state
pass in quick on xl0 proto udp from any to v.w.x.y/w port = 53 keep state
pass out quick on xl0 proto tcp/udp from v.w.x.y/w to any keep state
pass out quick on xl0 proto icmp from v.w.x.y/w to any keep state
pass out quick on xl0 proto tcp/udp from 10.0.0.0/24 to any keep state
pass out quick on xl0 proto icmp from 10.0.0.0/24 to any keep state
block in log quick on xl0 all

The above ruleset will permit you to run your own DNS, web server and mail servers, and block all other incoming traffic. All internal traffic is permitted to leave, so it is not a restrictive firewall from the users point of view on the inside, however not much from the outside will be permitted inside. The last 2 permit lines that specify 10.0.0.0/24 are only needed if you plan on running ipnat to allow the machines with 10. ip's through your firewall. Replace the 10.0.0.0/24 with whatever private ip scheme you are using.

References

Deviation v.1 locking down Linux/BSD 2001
reported by: sil
http://www.antioffline.com/deviation/ind.html

OpenBSD Home page
http://www.openbsd.org

Man Pages
IPF, File Formats
http://www.openbsd.org/cgi-bin/man.cgi?query=ipf&apropos=0&sektion=5&manpath=OpenBSD+Current&arch=i386&format=html

IPF, Maintenance Commands

http://www.openbsd.org/cgi-bin/man.cgi?query=ipf&apropos=0&sektion=8&manpath=OpenBSD+Current&arch=i386&format=html

IPNat, File Formats
http://www.openbsd.org/cgi-bin/man.cgi?query=ipnat&apropos=0&sektion=5&manpath=OpenBSD+Current&arch=i386&format=html

IPNat, Maintenance Commands
http://www.openbsd.org/cgi-bin/man.cgi?query=ipnat&apropos=0&sektion=8&manpath=OpenBSD+Current&arch=i386&format=html

IPF Official Howto Document
http://www.obfuscation.org/ipf/ipf-howto.txt

Example files
Filename: stable-ports.update
Location: Anywhere, just remember where it is
Command: **cvsup –g –L 2 stable-ports.update**

```
##################################
# stable-ports.update
*default release=cvs
*default delete use-rel-suffix
*default umask=002
*default host=cvsup.uk.OpenBSD.org
*default base=/cvs
*default prefix=/cvs
#  If your network link is a T1 or faster, comment out the following line.
*default compress
OpenBSD-src
OpenBSD-ports
```

Filename: ipf.rules
Location: /etc/ipf.rules
Command: Normally run on bootup, but the rules can be updated by the with the following command after you have changed the rules so no rebooting is required **ipf –Fa –f /etc/ipf.rules**

```
##########################################
# Firewall Ruleset
##########################################
# Drop non routable private addresses
block in quick on xl0 from 192.168.0.0/16 to any
block in quick on xl0 from 10.0.0.0/8 to any
block in quick on xl0 from 172.16.0.0/12 to any
```

```
block in quick on xl0 from 127.0.0.0/8 to any
block in quick on xl0 from 0.0.0.0/8 to any
block in quick on xl0 from 192.0.2.0/24 to any
block in quick on xl0 from 204.152.64.0/23 to any
block in quick on xl0 from 224.0.0.0/3 to any
#########################################
# Drop traffic with your own address
# space on outside interface
block in quick on xl0 from v.w.x.y/z to any
#########################################
# Permit traffic into network on specific
# ports for those services, add as needed
pass in quick on xl0 proto tcp from any to v.w.x.y/z port = 25 keep state
pass in quick on xl0 proto tcp from any to v.w.x.y/z port = 80 keep state
pass in quick on xl0 proto tcp from any to v.w.x.y/z port = 53 flags S keep state
pass in quick on xl0 proto udp from any to v.w.x.y/w port = 53 keep state
#########################################
# Permit all traffic to leave your internal
# netwrok
pass out quick on xl0 proto tcp/udp from any to any keep state
pass out quick on xl0 proto icmp from any to any keep state
#########################################
# Drop all traffic coming in that is not permitted
block in log quick on xl0 all
```

Last Updated: July 24th, 2017

Upcoming Training

SANS Prague 2017	Prague, Czech Republic	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS Boston 2017	Boston, MA	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS New York City 2017	New York City, NY	Aug 14, 2017 - Aug 19, 2017	Live Event
SANS Salt Lake City 2017	Salt Lake City, UT	Aug 14, 2017 - Aug 19, 2017	Live Event
Community SANS Omaha SEC401*	Omaha, NE	Aug 14, 2017 - Aug 19, 2017	Community SANS
Community SANS Trenton SEC401	Trenton, NJ	Aug 21, 2017 - Aug 26, 2017	Community SANS
Virginia Beach 2017 - SEC401: Security Essentials Bootcamp Style	Virginia Beach, VA	Aug 21, 2017 - Aug 26, 2017	vLive
SANS Chicago 2017	Chicago, IL	Aug 21, 2017 - Aug 26, 2017	Live Event
SANS Virginia Beach 2017	Virginia Beach, VA	Aug 21, 2017 - Sep 01, 2017	Live Event
SANS Adelaide 2017	Adelaide, Australia	Aug 21, 2017 - Aug 26, 2017	Live Event
Community SANS Pasadena SEC401 @ NASA	Pasadena, CA	Aug 23, 2017 - Aug 30, 2017	Community SANS
Mentor Session - SEC401	Minneapolis, MN	Aug 29, 2017 - Oct 10, 2017	Mentor
SANS San Francisco Fall 2017	San Francisco, CA	Sep 05, 2017 - Sep 10, 2017	Live Event
SANS Tampa - Clearwater 2017	Clearwater, FL	Sep 05, 2017 - Sep 10, 2017	Live Event
Mentor Session - SEC401	Edmonton, AB	Sep 06, 2017 - Oct 18, 2017	Mentor
SANS Network Security 2017	Las Vegas, NV	Sep 10, 2017 - Sep 17, 2017	Live Event
Mentor Session - SEC401	Ventura, CA	Sep 11, 2017 - Oct 12, 2017	Mentor
Community SANS Albany SEC401	Albany, NY	Sep 11, 2017 - Sep 16, 2017	Community SANS
Community SANS Columbia SEC401	Columbia, MD	Sep 18, 2017 - Sep 23, 2017	Community SANS
Community SANS Dallas SEC401	Dallas, TX	Sep 18, 2017 - Sep 23, 2017	Community SANS
SANS London September 2017	London, United Kingdom	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS Baltimore Fall 2017	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS Copenhagen 2017	Copenhagen, Denmark	Sep 25, 2017 - Sep 30, 2017	Live Event
Community SANS Boise SEC401	Boise, ID	Sep 25, 2017 - Sep 30, 2017	Community SANS
Baltimore Fall 2017 - SEC401: Security Essentials Bootcamp Style	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	vLive
Community SANS New York SEC401	New York, NY	Sep 25, 2017 - Sep 30, 2017	Community SANS
Rocky Mountain Fall 2017	Denver, CO	Sep 25, 2017 - Sep 30, 2017	Live Event
Community SANS Sacramento SEC401	Sacramento, CA	Oct 02, 2017 - Oct 07, 2017	Community SANS
SANS DFIR Prague 2017	Prague, Czech Republic	Oct 02, 2017 - Oct 08, 2017	Live Event
Community SANS Charleston SEC401	Charleston, SC	Oct 02, 2017 - Oct 07, 2017	Community SANS
Mentor Session - SEC401	Arlington, VA	Oct 04, 2017 - Nov 15, 2017	Mentor