OpenBSD

Creating the Ultimate Home Firewall
and Intrusion Detection System…
in under an hour

Derek J. Hunt
djhunt@uberh4x0r.org
http://www.uberh4x0r.org/projects

Presented at the
Rochester Area Linux Users Group
http://www.k-lug.org
July 10th, 2001
Version 0.1

Document Scope

This document is intended as a quick and easy walkthrough for setting up a firewall with NPAT, and Intrusion Detection capabilities under OpenBSD. The goal is, you should be up and running in under an hour even if you have never used OpenBSD.

OpenBSD is very portable and runs on Intel, Alpha, PowerPC, SPARC and some additional architectures. I used the June 1st 2.9 (i386) release as my testing platform. The OpenBSD configuration information is very similar to NetBSD and FreeBSD, and may in fact work on those platforms as well.

IPF is a very portable firewall toolkit, and also runs on numerous platforms. I believe these rules will work on any platform **`ipf`** supports, but they have not been tested.

This document is NOT a complete security guide, nor is it a thorough guide to IPF. If you want to dig a little deeper, please see the related documents section.

Obtaining and Installing OpenBSD

If you do not have a copy of OpenBSD, please consider ordering one (`http://www.openbsd.org/orders.html`). The OpenBSD project is funded through CD, and merchandising sales. The OpenBSD project tends to keep everyone (including the Linux advocates) in check. They set a level of excellence and dedication that is difficult to match.

The development model also proves that security can be achieved, and releases need not suffer from missed deadlines (The project releases a new version roughly, every six months).

If you insist on rolling your own, you can mirror the files from `ftp://ftp.openbsd.org`. You will need the files located in the **`2.9/i386`** directory. Grab the **`INSTALL.i386`** file and read through it carefully. It will explain the options for installation. If you are on a broadband, DS1 or any respectable connection, try the FTP installation. It is fast, and you can be up and running in a relatively short amount of time.

Creating a bootable CD

Well if you are too cheap to buy a copy, you can make your own bootable CD (Please keep in mind, that Theo owns the layout on the official CD, and it is copyrighted to entice people to purchase the CD directly).

I created the bootable CD from a RedHat 7.1 Linux machine. I mirrored the files from `ftp://ftp.openbsd.org/pub/OpenBSD/2.9/i386`.

In addition to mirroring all of the files located in the **`i386`** directory, I chose to copy the files from the **`2.9`** directory as well. The **`2.9`** directory contains the source code, and **`ports.tar.gz`**. I felt it would make a more complete CD with these files.

Once the files were downloaded, I used the **`mkisofs`** command to create the CD image.

```
mkisofs -l -L -v -r -T \
-V "OpenBSD-2.9" \
-A "OpenBSD v2.9-Release, Custom ISO, 06-05-2001, Please support \
OpenBSD http://www.openbsd.org/orders.html" \
-b 2.9/i386/cdrom29.fs \
-c boot.catalog \
-o openbsd-2.9.i386.iso \
-x openbsd-2.9.i386.iso ./OpenBSD/
```

Although this is not a true ISO image due to the long filenames, it contains all of the information to be usable under any UNIX OS as well as Microsoft Windows.

The ftp://ftp.openbsd.org/pub/OpenBSD/2.9/packages/i386 directory contains binary packages of popular software programs (from the **ports.tar.gz** collection). OpenBSD is pretty bare when you first get it installed (~120 Megs), and if you are a Linux user, many of your favorite utilities (i.e. **bash**, **pico** et cetera) will not be present. I usually make a second disc containing all of the packages (~650 Megs). This saves me from downloading this data later.

Once again, I used the **mkisofs** command to create the CD image.

```
mkisofs -l -L -v -r -T \
-V "OpenBSD-2.9" \
-A "OpenBSD v2.9-Release-Packages, Custom ISO, 06-05-2001, \
Please support OpenBSD http://www.openbsd.org/orders.html" \
-o openbsd-2.9-packages.i386.iso \
-x openbsd-2.9-packages.i386.iso ./OpenBSD/packages
```

Once `mkisofs` had finished both image files, I burned them to CD. This gives a complete OpenBSD 2.9 for i386 disc set.

Installing OpenBSD
Installing from CD is relatively easy, basically put the CD in, reboot and follow the instructions. Although it is not as flashy, it is very straightforward and you can be up and running in less than 15 minutes.

Installing from FTP is also easy. Download
ftp://ftp.openbsd.org/pub/OpenBSD/2.9/i386/floppy29.fs,
and **dd** (or **rawrite.exe**) it to a floppy. In the event that your network card does not work, try the **floppyB29.fs** and **floppyC29.fs** images. These images contain different drivers (for SCSI, RAID, Ethernet and PCMCIA devices). The **cdrom29.fs** image is actually a 2.88MB floppy image (if you have a 2.88MB Floppy Drive and a Working Disk).

If you are going to run **dd** from a Linux machine, run this command:
```
dd if=floppy29.fs of=/dev/fd0
```

If you need detailed instructions, check out: http://www.openbsd.org/faq/faq4.html.

A few things to keep in mind when installing OpenBSD:

- OpenBSD does **NOT** support as many devices as Linux. Checkout http://www.openbsd.org/i386.html for more information on supported devices.

- Some network cards need to be at a specific I/O address if you intend on installing via ftp. Knowing this in advance will save you hours of painful headaches and screaming obscenities.

Post installation tasks and preparing your system for NPAT, firewalling, and Intrusion Detection

Once you have OpenBSD installed, login to the system as root and check out **afterboot(8)** by typing:
```
man afterboot
```

Afterboot(8) explains some of the basic functionality of the system, and gives you some things to check out after the first complete boot.

1. **Edit the /etc/sshd_config file**
   Change the **PermitRootLogin entry** to **no** (The text should look like: **PermitRootLogin no**)
   This prevents users from logging into the system via **ssh** as root. As a general rule of thumb, you should **su –** to the superuser account via a normal user.

2. **Add a general user to the system**
   Run the **adduser** command. Follow the instructions for setting up default entries.
   OpenBSD prevents users from using **su** unless they are in the **wheel** group. When the **adduser** command asks you if you would like to invite this user to other groups, type **wheel** and hit enter.

3. **Edit the /etc/sysctl.conf file**
   Uncomment the **`net.inet.ip.forwarding=1`** entry.
   This permits the machine to forward (route) packets.
   Linux handles this in a similar fashion (**`echo 1 > /proc/sys/net/ipv4/ip_forward`**).

4. **Edit the /etc/rc.conf file**
   Change the **`ipnat`** and **`ipfilter`** entries to **`YES`**
   example: **`ipnat=YES`**
   example: **`ipfilter=YES`**

Reboot your system to allow the changes to take effect, and your system is now ready for firewalling.

IP Filter and IPNat

IP Filter is very robust package that allows for complete control over the entire filtering stack. Coupled with the secure nature of OpenBSD, ipf is a great choice for a small, efficient router/firewall.

Ipf is part of some recent headaches in the BSD world. A recent change in the license prompted Theo to remove it from the OpenBSD distribution. This means that in the future, you may have to download it from `http://coombs.anu.edu.au/~avalon/`.

Linux users will notice a big difference in the methodology of `ipf` and `ipchains`/`iptables`. `ipchains` and `iptables` process rules by running a new command each time:

Example:**`/sbin/ipchains -A user_msq -s 0/0 -d 0/0 -j MASQ`**
`/sbin/ipchains -A forward -s 192.168.0.0/24 -d 0/0 -i eth0 -j user_msq`

Example: **`/sbin/iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE`**

`ipf` and **`ipnat`** use configuration files (**`/etc/ipf.rules`** and **`/etc/ipnat.rules`**). Some things to keep in mind about these files:

- One rule per line
- "**#**" like most UNIX configuration files, denotes a comment.
- Whitespace is ignored.
- Rules are processed from Top to Bottom.
- The last matching rule is the one that counts!
  This means if you block everything:
  `block in all`
  and later open it up with:
  `pass in all`
  Every packet will go through.

Setting up /etc/ipnat.rules
/etc/ipnat.rules contains the configuration for running Network Address Translation (NAT) and Network and Port Address Translation (NPAT) rules.

To get basic NAT running you will need three lines in your **`/etc/ipnat.rules`** file.
The format for these lines are:

```
map <external device> <reserved subnet>/<netmask> -> <external device>/<netmask> proxy port ftp ftp/tcp
map <external device> <reserved subnet>/<netmask> -> <external device>/<netmask> portmap tcp/udp 10000:20000
map <external device> <reserved subnet>/<netmask> -> <external device>/<netmask>
```

In the following example **sf3** is the device connected to the Internet (the external device).
192.168.0.0/24 is the reserved subnet I am providing NAT services for.

```
map sf3 192.168.0.0/24 -> sf3/32 proxy port ftp ftp/tcp
map sf3 192.168.0.0/24 -> sf3/32 portmap tcp/udp 10000:20000
map sf3 192.168.0.0/24 -> sf3/32
```

Let me explain what these rules do. The first rule,
`map sf3 192.168.0.0/24 -> sf3/32 proxy port ftp ftp/tcp`
allows ftp to work correctly through the NAT/firewall. Ftp is a strange beast, and has always been an issue with proxies and firewalls. **`ipf`** takes care of these issues by creating an application proxy which basically keeps a careful eye on the ftp session. What this means to is, you do not have to set passive mode for your ftp sessions. Linux accomplishes this with a kernel module and `ipchains` rules.

The second rule,
`map sf3 192.168.0.0/24 -> sf3/32 portmap tcp/udp 10000:20000`
maps many addresses from your reserved subnet (**`192.168.0.0/24`**) to the address on your external device (**`sf3`**). This is essentially what masquerading does for Linux. This also maps all outgoing ports to a high port range (**`10000-20000`**).

The third rule,
`map sf3 192.168.0.0/24 -> sf3/32`
creates a general IP mapping many addresses to one.

Note: *If you are running a dynamic address (DHCP, PPP, or a VPN connection), you can substitute the second external device entry (listed as* **`sf3/32`** *above) with* **`0/32`**. **`ipnat`** *is smart enough to figure out the address by accessing the device from the first entry.*

Port forwarding with IPNat
Port forwarding is called redirection in ipnat. Redirection is established with the **rdr** keyword.
`rdr <external device> <external address>/<netmask> port <port number> -> <internal address> port <port number>`
For example, say I want to forward **port 80** (http) from my live address (**206.9.88.129**) to a webserver located on protected subnet (**192.168.0.1**). I would put the following command in my **/etc/ipnat.rules** file:
`rdr sf3 206.9.88.129/32 port 80 -> 192.168.0.1 port 80`

Firewalling with IP Filter
The filter rules are stored in the configuration file **`/etc/ipf.rules`**. The actual rule construction is a very detailed. I started by blocking all reserved addresses, adding some anti-spoofing rules, and some filters for remote access to some select services on the machine (SSH and HTTPS). I did some research on the OpenBSD mailing lists and the IP Filter mailing list and found Mike (*sdnetorg@yahoo.com*) had a fairly complete rule set that he had compiled from the IP Filter mailing list. My rule set expands on his compiled list, and enables NPAT. This list is by no means the "*be all, end all*" for securing your machine, but it will offer you reasonable security, with very little maintenance.

I will not explain each rule in detail, instead I will explain them in "blocks". These blocks cover rules with similar functionality. If you want a thorough IP Filter HOW-TO, please visit http://www.obfuscation.org/ipf/. Brenden Conoboy (synk@swcp.com) and Erik Fichtner (emf@obfuscation.org) have written an excellent guide to IP Filter, this documents served as my main guide when I started creating the rules.

I begin the document by explaining the interfaces, and they act as a reference. The device **`sf3`** is connection to the internet with the IP address **`206.9.88.129`** . This happens to be the forth port in an Adaptec 4 Port Ethernet NIC. The device, **`fxp0`**, is an on-board Intel Ethernet card. This card has the reserved IP address **`192.168.0.254`** .

```
##################################################################
# Interface information
# sf3 is connected to the internet - EXTERNAL
# fxp0 is connected internally - INTERNAL
##################################################################
```

The next block contains some rules that block and log malformed packets that are generally too small to actually exist with a real payload. This means that these packets come from Denial of Service (DoS) type tools, or strange remote TCP/IP stack exploits. It is generally a good idea to block these.

```
#####################################################################
# Hmmm you guys are too small to be real, are you trying to crash
# my box?
#####################################################################
block in log quick all with short
block in log quick all with opt lsrr
block in log quick all with opt ssrr
block in log quick all with ipopts
block in log quick on sf3 all with frags
```

This block covers the network utility NMAP (http://www.insecure.org). NMAP is used to scan for open ports, and is often the first tool a system cracker will use in probing a system. These rules are designed to identify, and mess with NMAP.

```
#####################################################################
# Hi there NMAP
#####################################################################
block in log quick on sf3 proto tcp from any to any flags FUP
block in log quick on sf3 proto tcp from any to any flags SF/SFRA
block in log quick on sf3 proto tcp from any to any flags /SFRA
```

This block sets up the three interfaces that are present on the machine (**sf3**, **fxp0**, and the loopback device, **lo0**). The rules for loopback allow everything to pass-through if it is on a loopback device.
The next block sets up the external interface (**sf3**) to block everything. This includes traffic coming in as well as going out. The third block sets up the internal interface (fxp0) to block all traffic as well.
The default rule for our firewall is: **BLOCK EVERYTHING,** and take names later. Linux users will recognize this as the DENY rule for ipchains. It is a good rule of thumb to deny everything and then explicitly allow access to services. Its a lot easier put rules in then it is to track down some strange service that installs something new.

These rules also introduce the concept of groups. Groups are basically like naming your rules under ipchains (USER_MASQ et cetera). We represent these group by the number 100, 150, 200, 250. The cover the external and internal devices incoming and outgoing traffic.

```
#####################################################################
# Handle everything from loopback, I am considering trusted
#####################################################################
pass in      quick on lo0 all
pass out     quick on lo0 all

#####################################################################
# Setup rules for interface: sf3 (EXTERNAL)
#####################################################################
block in on sf3 all head 100  # INCOMING
block out on sf3 all head 150 # OUTGOING

#####################################################################
# Setup rules for interface: fxp0 (INTERNAL)
#####################################################################
block in on xl0 all head 200  # INCOMING
block out on xl0 all head 250 # OUTGOING
```

The next block, and it's a big one, contains a list of IPs that you will want to block. These cover reserved addresses, the Sun Cluster, and Class D & E blocks. Generally these IPs should NEVER show up on a live interface as they are NOT ROUTABLE addresses. If you notice these IPs, chances are someone is spoofing them. We group these rules in the 100 group, which is the external interface's incoming traffic.

```
#################################################################
# Block packets coming from anything on the internet that
# shouldn't be there
# whois.arin.net
#################################################################
block in log quick from 0.0.0.0/7 to any group 100
block in log quick from any to 0.0.0.0/7 group 100
…
block in log quick from 240.0.0.0/4 to any group 100 #Unspecified (Class >D)
block in log quick from any to 240.0.0.0/4 group 100 #Unspecified (Class >D)
```

This block contains rules pertaining to our **ALLOWABLE** incoming traffic. Since I am only allowing SSH and HTTPS traffic, only ports 22 and 443 are available. Below that are the incoming ICMP rules. If you want extra security, comment all of them. This effectively shuts off ALL ICMP traffic coming to your box. This means, people will not be able to **`traceroute`** to you, or even ping you. In the rule set below, I am allow Echo Reply (I am accepting pings) and I and sending a **TCP RST** to each connection that does an inquiry to one of the other ICMP flags. This means that I only accept pings, and all others will get a *Destination Unreachable*, rather than a *Request Timed Out*.

```
#################################################################
# Incoming traffic on sf3
#################################################################

# HTTPS/SSL
pass in quick proto tcp from any to any port = 443 flags S/SA keep state group 100
# SSH
pass in quick proto tcp from any to any port = 22 flags S/SA keep state group 100

# allow certain classes of ICMP
# Traceroute Unix requires type: 3,  UDP port > 33000
# Tracert Microsoft requires type: 0, 8, 11
# Ping (Packet InterNet Groper) requires type: 8, 0
#pass in quick proto icmp all group 100
pass in quick proto icmp all icmp-type 0 group 100 #Echo Reply
#pass in log quick proto icmp all icmp-type 3 group 100 #Destination Unreachable
#pass in log quick proto icmp all icmp-type 4 group 100 #Source Quench
#pass in log quick proto icmp all icmp-type 5 group 100 #Redirect
#pass in log quick proto icmp all icmp-type 8 group 100 #Echo Request
#pass in log quick proto icmp all icmp-type 11 group 100 #Time Exceeded
#pass in log quick proto icmp all icmp-type 12 group 100 #Parameter Problem
#pass in log quick proto icmp all icmp-type 13 group 100 #Timestamp request
#pass in log quick proto icmp all icmp-type 14 group 100 #Timestamp reply
#pass in log quick proto icmp all icmp-type 15 group 100 #Information Request
#pass in log quick proto icmp all icmp-type 16 group 100 #Information Reply
#pass in log quick proto icmp all icmp-type 17 group 100 #Address Mask Request
#pass in log quick proto icmp all icmp-type 18 group 100 #Address Mask Reply
block in quick proto icmp all group 100
# if nothing applies, block and return icmp-replies (unreachable and rst)
block return-icmp(net-unr) in log proto udp from any to any group 100
block return-rst in log proto tcp from any to any group 100
```

Now we need to allow our machine to access the net, remember the default policy was to deny all traffic. These next rules allow the firewall to send outbound traffic. I set these rules to allow any and all traffic to go out.

```
#####################################################################
# Outgoing traffic on sf3
#####################################################################
# I know, I know, but I am considering the firewall to be trusted
pass out quick proto tcp all keep state group 150
pass out quick proto udp all keep state group 150
pass out quick proto icmp all keep state group 150
```

The last block contains rules that allow our internal address scheme (192.168.0.0/24) to send traffic in and out.

```
#####################################################################
# Incoming traffic on fxp0 (INTERNAL)
#####################################################################
pass in quick from 192.168.0.0/24 to any  group 200

#####################################################################
# Outgoing traffic on fxp0 (INTERNAL)
#####################################################################
pass out quick from any to any group 250
```

Configuring these rules for your system
These rules are available for download at http://www.uberh4x0r.org/download/derek/openbsd-fw.tar.gz. To use these rules, edit the files accordingly. If you are feeling brave you can type them in manually and discard rules that you feel you don't need, or are not applicable to your situation.

Once your rules are in place, run :
sh –x /etc/netstart

This command restart your network services (similar to **/etc/init.d/network restart**). Once you are done, you should be able to access the Internet under a fully functional OpenBSD PNAT Firewall.

Intrusion Detection Made Simple

There are number of different Intrusion Detection Systems (IDS), and each has its own set of unique advantages/disadvantages, and choosing an IDS can be a tough decision. Many of these decisions are based on the network architecture and the type of IDS (Host based or network based). IDS design and implementation is well beyond the scope of this document. Because of this, a few liberties have been taken in choosing the IDS.

- It is assumed that the experience and expertise of the user is at a novice level.
- It is assumed that the user will have all services shut off externally and is unlikely to have a host exploit.

These two reasons pushed me towards selecting a Network IDS. In my personal experience, I tend to receive network attacks rather than host exploits. This is mainly due to the fact that I do not allow external access to my firewall. The only user accounts are my own, and I have all methods of remote access shut off (Although I do run a host intrusion detection system out of habit).

Of the common Network Intrusion Detection Systems out there, Snort (http://www.snort.org) is perhaps the popular. It is highly portable (running on nearly every UNIX as well as Windows 2000) and receives updated IDS signatures almost everyday. This means that the IDS is generally up to date and ready to catch these attacks shortly after they are released into the wild.

In addition to the active development, Snort has a very active accessory community. This means that people are writing applications to help Snort fit into different environments, increase its usability and functionality. For example, there is a utility that runs on Linux systems that will add new ipchains rules when an attack signature has been detected.

The popularity of Snort, has led to some fantastic log analysis tools. One of the more popular log analyzers, SnortSnarf (http://www.silicondefense.com/software/snortsnarf/index.htm) takes a thorough look through the snort log directory and generates very useful HTML output. SnortSnarf also puts links to the arachnids and CVE attack databases and it provides a fairly detailed summary of the attack.

Installing Snort

If you have the packages CD, you can copy the file **snort-1.7.tgz** from there, or you can ftp the file from ftp://ftp.openbsd.org/pub/OpenBSD/2.9/packages/i386/snort-1.7.tgz.
Once you have the file downloaded, install it with **pkg_add** .
Example: **pkg_add ./snort-1.7.tgz**

After Snort is installed, I create the directory **/etc/snort** to store the Snort configuration files. It is a good idea to download the latest Snort rules from http://www.snort.org/Files/Current/snortrules.tar.gz. Once this file is downloaded, you can extract the archive into **/etc/snort**.
Example:

```
cd /etc/snort; tar zxvf ~/snortrules.tar.gz ; \
cat snort.conf | sed 's|include |include /etc/snort/|g' > snort.conf; \
cd ~
```

This command will extract the current Snort rules into **/etc/snort** and adjust the include paths.

Next you will need to edit the /etc/snort/snort.conf file. This file contains all of the configuration information for Snort.

Find the entry labeled: **var HOME_NET**.
The format for this entry is: **var HOME_NET <external address>/<netmask>**
You will need to change this to fit your external IP address.
Example:

```
var HOME_NET 206.9.88.129/32
```

Next, search for the entry labeled var DNS_SERVERS.
The format for this entry is the same as HOME_NET,

```
var DNS_SERVERS [<external address>/<netmask>,…]
```

Change this entry to match your DNS servers.
Example:

```
var DNS_SERVERS [206.9.64.100/32, 206.9.106.180/32]
```

Save and close the file.

Create the directory **/var/log/snort**
This is the default directory where Snort will log alerts.

Snort is now configured for basic operations.

To start snort, run the following command

```
/usr/local/bin/snort -c /etc/snort/snort.conf -i <external device> -D
```

Example: **/usr/local/bin/snort -c /etc/snort/snort.conf -i sf3 –D**

Adding this command to the **/etc/rc.local** file will start snort automatically when the system restarts.

Snort is now running in daemon mode. Now I know I will receive some criticism for snort running as root, this can be an issue. The ideal way is to create a snort user, remove shell access, and restrict the log directory for the snort user. You can certainly do this, and snort allows you to select a specific user and group to run as. This is a required feature in a highly secure environment. If you would like to learn more about Snort, please visit http://www.snort.org and read through the faqs, they will explain everything you need to know.

Installing SnortSnarf

First obtain SnortSnarf from http://www.silicondefense.com/software/snortsnarf/index.htm.

The release used in this test was 052301.

Uncompress and untar the archive and change into the SnortSnarf Directory.

SnortSnarf requires Time::JulianDay. If your system does not have this you can install from CPAN or install it from the 'Time-modules' sub-directory (included with SnortSnarf).

There is not an installer included with SnortSnarf, I generally copy this directory to /usr/local and create a symbolic link to snort snarf.

Example: **`mv SnortSnarf-052301.1 /usr/local/ ; cd /usr/local/ ; ln –s SnortSnarf-052301.1 snortsnarf`**

SnortSnarf creates an html report that can be viewed locally or on your webserver. I created a batch file to run the output to a directory in my webserver htdocs directory:

```
#!/bin/sh
# report.sh – Run a snortsnarf report
cd /usr/local/snortsnarf/
./snortsnarf.pl -rulesfile /etc/snort/snort.conf -dns -d /var/www/htdocs/snort
/var/log/snort/alert
cd ~
```

SnortSnarf has several dependencies within it's directory structure. Unless you install the perl dependencies, it's a good idea to run the report from a shell script.

The above report references the snort rules stored in **`/etc/snort/snort.conf`**, stores the output in **`/var/www/htdocs/snort`**, and reads from the alert file located in **`/var/log/snort`**.

I generally run the report script once every hour through **`cron,`** but bear in mind SnortSnarf is very memory intensive while it is running. On a PIII 800 MHz, with an alert directory around 15 megs it takes a couple minutes. In constrat on a Pentium 133MHz, the same report takes almost an hour. There are several factors to take into consideration for the report, disk I/O, memory, and processor need to be taken into consideration when figuring out your frequency to run the report.

Related Documents and websites

This HOW-TO and related files can be downloaded from http://www.uberh4x0r.org/projects .

Software Used to create the firewall:
OpenBSD (Operating System) - http://www.openbsd.org
Snort (Intrusion Detection System) – http://www.snort.org
SnortSnarf (IDS reporting tool) - http://www.silicondefense.com/software/snortsnarf/index.htm
IPF (Packet Filtering, Firewall Tool) - http://coombs.anu.edu.au/~avalon/
IP Filter HOW-TO, please visit http://www.obfuscation.org/ipf/

General security sites:
SecurityFocus – Great Informational site, complete with forums, reviews, and technical how-to guides. Has a great Linux and Intrusion Detection Section. - http://www.securityfocus.com/

BugTraq – THE definitive, full disclosure security mailing list. – To subscribe, visit:
http://www.securityfocus.com/about/feedback/subscribe.html

Security Portal – Another great security related information site – http://www.securityportal.com

PacketStorm Security – A tremendous archive of hacking/cracking tools, text and information. -
http://packetstorm.securify.com/

Contents of my /etc/ipnat.rules file.

```
# $OpenBSD: ipnat.rules,v 1.2 1999/05/08 16:33:10 jason Exp $
#
# See /usr/share/ipf/nat.1 for examples.
# edit the ipnat= line in /etc/rc.conf to enable Network Address Translation

#map ppp0 10.0.0.0/8 -> ppp0/32 portmap tcp/udp 10000:20000

# NAT for the 192.168.0.0/24 subnet
map sf3 192.168.0.0/24 -> sf3/32 proxy port ftp ftp/tcp
map sf3 192.168.0.0/24 -> sf3/32 portmap tcp/udp 10000:20000
map sf3 192.168.0.0/24 -> sf3/32
```

Contents of my /etc/ipf.rules file.

```
#       $OpenBSD: ipf.rules,v 1.6 1997/11/04 08:39:32 deraadt Exp $
#
# IP filtering rules.  See the ipf(5) man page for more
# information on the format of this file, and /usr/share/ipf
# for example configuration files.
#
# Pass all packets by default.
# edit the ipfilter= line in /etc/rc.conf to enable IP filtering
#
#pass in from any to any
#pass out from any to any

######################################################################
# Interface information
# sf3 is connected to the internet - EXTERNAL
# fxp0 is connected internaly - INTERNAL
######################################################################

######################################################################
# Hmmm you guys are too small to be real, are you trying to crash
# my box?
######################################################################
block in log quick all with short
block in log quick all with opt lsrr
block in log quick all with opt ssrr
block in log quick all with ipopts
block in log quick on sf3 all with frags

######################################################################
# Hi there NMAP
######################################################################
block in log quick on sf3 proto tcp from any to any flags FUP
block in log quick on sf3 proto tcp from any to any flags SF/SFRA
block in log quick on sf3 proto tcp from any to any flags /SFRA

######################################################################
# Handle everything from loopback, I am considering it trusted
######################################################################
pass in      quick on lo0 all
pass out     quick on lo0 all

######################################################################
# Setup rules for interface: sf3 (EXTERNAL)
######################################################################
block in on sf3 all head 100  # INCOMING
block out on sf3 all head 150 # OUTGOING

######################################################################
# Setup rules for interface: fxp0 (INTERNAL)
######################################################################
block in on xl0 all head 200  # INCOMING
block out on xl0 all head 250 # OUTGOING
######################################################################
# Block packets coming from anything on the internet that
# shouldn't be there
# whois.arin.net
######################################################################
block in log quick from 0.0.0.0/7 to any group 100
block in log quick from any to 0.0.0.0/7 group 100
block in log quick from 0.0.0.0/8 to any group 100 #Odd Loopback Reserved
block in log quick from any to 0.0.0.0/8 group 100 #Odd Loopback Reserved
block in log quick from 2.0.0.0/8 to any group 100 #Unassigned
block in log quick from any to 2.0.0.0/8 group 100 #Unassigned
block in log quick from 5.0.0.0/8 to any group 100 #Unassigned
block in log quick from any to 5.0.0.0/8 group 100 #Unassigned
block in log quick from 10.0.0.0/8 to any group 100 #Private Class A RFC 1918
block in log quick from any to 10.0.0.0/8 group 100 #Private Class A RFC 1918
```

```
block in log quick from 20.20.20.0/24 to any group 100 # Netblock reserved by Sun Microsystems for
                                                       # Private Cluster Interconnect
block in log quick from any to 20.20.20.0/24 group 100 # Netblock reserved by Sun Microsystems for
                                                       # Private Cluster Interconnect
block in log quick from 23.0.0.0/8 to any group 100
block in log quick from any to 23.0.0.0/8 group 100
block in log quick from 27.0.0.0/8 to any group 100
block in log quick from any to 27.0.0.0/8 group 100
block in log quick from 31.0.0.0/8 to any group 100
block in log quick from any to 31.0.0.0/8 group 100
block in log quick from 67.0.0.0/8 to any group 100
block in log quick from any to 67.0.0.0/8 group 100
block in log quick from 68.0.0.0/6 to any group 100
block in log quick from any to 68.0.0.0/6 group 100
block in log quick from 72.0.0.0/5 to any group 100
block in log quick from any to 72.0.0.0/5 group 100
block in log quick from 80.0.0.0/4 to any group 100
block in log quick from any to 80.0.0.0/4 group 100
block in log quick from 96.0.0.0/3 to any group 100 #unassigned with the exception of 127.0.0.0/8
block in log quick from any to 96.0.0.0/3 group 100
block in log quick from 127.0.0.0/8 to any group 100 #Loopback
block in log quick from any to 127.0.0.0/8 group 100 #Loopback
block in log quick from 128.0.0.0/16 to any group 100
block in log quick from any to 128.0.0.0/16 group 100
block in log quick from 128.66.0.0/16 to any group 100
block in log quick from any to 128.66.0.0/16 group 100
block in log quick from 169.254.0.0/16 to any group 100 #assigned by the IANA for use in auto-
                                                         # configuration of DHCP default
block in log quick from any to 169.254.0.0/16 group 100 #assigned by the IANA for use in auto-
                                                         # configuration of DHCP default
block in log quick from 172.16.0.0/16 to any group 100 #Private Class B
block in log quick from any to 172.16.0.0/16 group 100 #Private Class B
block in log quick from 191.255.0.0/16 to any group 100
block in log quick from any to 191.255.0.0/16 group 100
block in log quick from 192.0.2.0/24 to any group 100 # been reserved for use as an example IP netblock
                                                      # for documentation authors
block in log quick from any to 192.0.2.0/24 group 100 # been reserved for use as an example IP netblock
                                                      # for documentation authors
block in log quick from 192.168.0.0/16 to any group 100 #Private Class C RFC 1918 Private
block in log quick from any to 192.168.0.0/16 group 100 #Private Class C RFC 1918 Private
block in log quick from 197.0.0.0/8 to any group 100 #Unassigned
block in log quick from any to 197.0.0.0/8 group 100 #Unassigned
block in log quick from 201.0.0.0/8 to any group 100
block in log quick from any to 201.0.0.0/8 group 100
block in log quick from 204.152.64.0/23 to any group 100 #Netblock reserved by Sun Microsystems for
                                                          #Private Cluster Interconnect
block in log quick from any to 204.152.64.0/23 group 100 #Netblock reserved by Sun Microsystems for
                                                          #Private Cluster Interconnect
block in log quick from 224.0.0.0/3 to any group 100  #Multicast Class D
block in log quick from any to 224.0.0.0/3 group 100  #Multicast Class D
block in log quick from 240.0.0.0/4 to any group 100 #Unspecified (Class >D)
block in log quick from any to 240.0.0.0/4 group 100 #Unspecified (Class >D)

#####################################################################
# Incoming traffic on sf3
#####################################################################

# HTTPS/SSL
pass in quick proto tcp from any to any port = 443 flags S/SA keep state group 100

# SSH
pass in quick proto tcp from any to any port = 22 flags S/SA keep state group 100

# allow certain classes of ICMP
# Traceroute Unix requires type: 3,  UDP port > 33000
# Tracert Microsoft requires type: 0, 8, 11
# Ping (Packet InterNet Groper) requires type: 8, 0
#pass in quick proto icmp all group 100
pass in quick proto icmp all icmp-type 0 group 100 #Echo Reply
#pass in log quick proto icmp all icmp-type 3 group 100 #Destination Unreachable
```

```
#pass in log quick proto icmp all icmp-type 4 group 100 #Source Quench
#pass in log quick proto icmp all icmp-type 5 group 100 #Redirect
#pass in log quick proto icmp all icmp-type 8 group 100 #Echo Request
#pass in log quick proto icmp all icmp-type 11 group 100 #Time Exceeded
#pass in log quick proto icmp all icmp-type 12 group 100 #Parameter Problem
#pass in log quick proto icmp all icmp-type 13 group 100 #Timestamp request
#pass in log quick proto icmp all icmp-type 14 group 100 #Timestamp reply
#pass in log quick proto icmp all icmp-type 15 group 100 #Information Request
#pass in log quick proto icmp all icmp-type 16 group 100 #Information Reply
#pass in log quick proto icmp all icmp-type 17 group 100 #Address Mask Request
#pass in log quick proto icmp all icmp-type 18 group 100 #Address Mask Reply
block in quick proto icmp all group 100
# if nothing applies, block and return icmp-replies (unreachable and rst)
block return-icmp(net-unr) in log proto udp from any to any group 100
block return-rst in log proto tcp from any to any group 100

#####################################################################
# Outgoing traffic on sf3
#####################################################################
# I know, I know, but I am considering the firewall to be trusted
pass out quick proto tcp all keep state group 150
pass out quick proto udp all keep state group 150
pass out quick proto icmp all keep state group 150

#####################################################################
# Incoming traffic on fxp0 (INTERNAL)
#####################################################################
pass in quick from 192.168.0.0/24 to any  group 200

#####################################################################
# Outgoing traffic on fxp0 (INTERNAL)
#####################################################################
pass out quick from any to any group 250
```