Table of Contents

1 Document Overview

Author: NETIKUS.NET ltd
Date: 21st Oct 2003
Revision: 1.1

OpenBSD Transparent Firewall Installation Guide

Title	OpenBSD Transparent Firewall Installation Guide
Summary	Transparent firewall installation with OpenBSD, using OpenBrick-E hardware as an example
Software	OpenBSD 3.3
Hardware	OpenBrick-E with CF card or generic i386 PC (http://www.hacom.net)
Skill Level	Beginner - Intermediate
Skills Required	- Basic understanding of Unix - Basic understanding of TCP/IP (firewall configuration) - Basic familiarity of Unix shell (e.g. csh, bash) - Basic usage of the vi editor
Acknowledgements	Thanks to Chuck Yerkes for offering tips on getting OpenBSD to work read-only.
Download	http://www.netikus.net/ (guides section)

2 Document Details

Overview

This document describes how to install a transparent firewall (based on OpenBSD) on an Openbrick-E hardware using a CompactFlash (512Mb or 256Mb) card. We will make a reasonable effort on installing OpenBSD mostly read-only.

The Openbrick-E comes with a VIA C3 processor, 256Mb of RAM and three Realtek NICs. Other models and variations are available (more RAM, different NICs) but we'll be referring to the default configuration in this document. For more information on available OpenBrick models please visit http://www.openbrick.org or the distributor in the US http://www.hacom.net/.

You can also use this document as a guideline to install OpenBSD on different types of hardware, read-write if you do have sufficient hard disk space.

Transparent Firewall

A transparent firewall is an ethernet bridge that transparently filters out potentially malicious packets. By not assigning any IP addresses to the two bridging interfaces the device is more or less invisible on the network. The 3rd NIC can be used to administer this machine using a private IP address.

Possible Applications

Transparent firewalls are useful in a variety of network scenarios since they do not require a reconfiguration of other networking equipment. They can be used to

- protect an entire network by attaching it to the main gateway (router)
- protect a subnet by attaching it to uplink hub ports
- protect a newly installed computer by attaching it between the computer and the hub

Why?

This guide was written after a transparent firewall was installed at a major university located in the United States. The firewall was needed to restrict access to certain computers, most notable unprotected Windows based servers and clients.

3 Setup (Openbrick Only)

We will install OpenBSD 3.3 (without Xfree) on a CF disk with 512Mb of size, this makes the installation easy and less complex since there are no space restrictions for a default installation. Feel free to install OpenBSD on smaller disks as well, we will give some hints on how to accomplish that later on.

3.1 BIOS Settings

Enter the BIOS and modify the following settings:

```
Change framebuffer to 2Mb
Change AGP aperture to 4Mb
Disable built-in USB
Disable built-in sound
Disable built-in modem
Disable built-in floppy
(a bug in the current BIOS version makes this setting ineffective)
```

3.2 Disabling Network Boot

By default the built-in network cards try to boot over the network which causes a significant boot delay. When you see the **SHIFT-F10** prompt press this combination (3x for each NIC) and disable this feature, save the changes with **F4**.

4 Installation

After inserting the bootable CD that you either downloaded or purchased through the openbsd.org website you will see a screen similar to the one shown below:

```
"Intel 82371AB Power Mgmt" rev 0x00 at pci0 dev 7 function 3 not configured
vga1 at pci0 dev 15 function 0 vendor "VMware", unknown product 0x405 rev 0x00
wsdisplay0 at vga1: console (80x25, vt100 emulation)
le1 at pci0 dev 16 function 0 "AMD 79c970 PCnet-PCI LANCE" rev 0x10
le1: address 00:0c:29:d2:28:2f
le1: 8 receive buffers, 2 transmit buffers
le1: interrupting at irq 11
"Ensoniq AudioPCI97" rev 0x02 at pci0 dev 17 function 0 not configured
isa0 at pcib0
isadma0 at isa0
pckbc0 at isa0 port 0x60/5
pckbd0 at pckbc0 (kbd slot)
pckbc0: using irq 1 for kbd slot
wskbd0 at pckbd0: console keyboard, using wsdisplay0
npx0 at isa0 port 0xf0/16: using exception 16
pccom0 at isa0 port 0x3f8/8 irq 4: ns16550a, 16 byte fifo
pccom1 at isa0 port 0x2f8/8 irq 3: ns16550a, 16 byte fifo
fdc0 at isa0 port 0x3f0/6 irq 6 drq 2
biomask c240 netmask ca40 ttymask ca42
rd0: fixed, 3560 blocks
wd0: no disk label
root on rd0a
rootdev=0x1100 rrootdev=0x2f00 rawdev=0x2f02
erase ^?, werase ^W, kill ^U, intr ^C, status ^T
(I)nstall, (U)pgrade or (S)hell? _
```

We can either install or upgrade an existing OpenBSD installation, in addition we can "misuse" the installation CD as a rescue CD and enter the shell by pressing S.

Answer the following questions as shown below. Default answers are always shown in brackets, simply pressing enter answer a question using this default.

```
Terminal type? [vt220] vt220
Do you wish to select a keyboard encoding table? [n] n
Proceed with install? [n] y
```

4.1 Partitioning

If the computer on which you are installing OpenBSD has one IDE harddisk (CF disk) then you should see the following screen:

```
You will now initialize the disk(s) that OpenBSD will use. To enable all
available security features you should configure the disk(s) to allow the
creation of separate filesystems for /, /tmp, /var, /usr, and /home.

Available disks are: wd0.
Which one is the root disk? (or 'done') [wd0] _
```

Since **wd0** is the only available disk we press enter and are presented with the question

```
Do you want to use *all* of wd0 for OpenBSD? [no] yes
```

which we answer with a yes since we are not installing any other Operating Systems. The OpenBSD installer will then create an OpenBSD partition on your harddisk and enter the disklabel editor **disklabel** automatically. You will see something like this:

```
# using MBR partition 3: type A6 off 63 (0x3f) size 2096577 (0x1ffdc1)

Treating sectors 63-2096640 as the OpenBSD portion of the disk.
You can use the 'b' command to change this.

Initial label editor (enter '?' for help at any prompt)
> p
device: /dev/rwd0c
type: ESDI
disk: ESDI/IDE disk
label: VMware Virtual I
bytes/sector: 512
sectors/track: 63
tracks/cylinder: 16
sectors/cylinder: 1008
cylinders: 2080
total sectors: 2097152
free sectors: 2096577
rpm: 3600

16 partitions:
#        size   offset    fstype   [fsize bsize   cpg]
  a:  2096577       63    unused        0     0
  c:  2097152        0    unused        0     0
> _
```

This shows two partitions in the lower area – **a** and **c** where **c** represents the entire partition. The number **2097152** is the total size of the disk in **512** byte blocks – exactly 1Gb (2097152 * 512 byte). We will remove the **a** slice that was created by the installer and setup our own partition instead.

```
  c:  2097152        0    unused        0     0
> d a
> a a
offset: [63]
size: [2096577] 65M
Rounding to nearest cylinder: 132993
FS type: [4.2BSD]
Mount point: [none] /
> a b
offset: [133056]
size: [1963584] 1M
Rounding to nearest cylinder: 2016
FS type: [swap]
> a d
offset: [135072]
size: [1961568] 65M
Rounding to nearest cylinder: 133056
FS type: [4.2BSD]
Mount point: [none] /var
> a e
offset: [268128]
size: [1828512]
FS type: [4.2BSD]
Mount point: [none] /usr
> _
```

To create the partitions **a, b, d** and **e** type what you see in the screenshot above.
First we delete the a partition by typing **d a**, then we add the **root /** partition, add the **swap** partition (very small on purpose) and finally add the **/var** and **/usr** partitions.

We used the default provided when you see no input next to the [] brackets.

We write our changes by typing

w
q

and proceed to creating the file systems.

If you are installing onto a **256Mb CF disk** then we recommend the following partition sizes:

- `/     50Mb`
- `swap  1Mb`
- `/var  25M`
- `/usr  rest (~180Mb)`

When installing onto a reasonably big hard drive please consult the OpenBSD manual (FAQ). We generally use a ~128Mb big root partition, a swap partition the size of RAM (or smaller), a /usr partition ~1024Mb, a 128Mb /var partition and, if possible, a separate /var/log partition that is even bigger.

Setup will now create all file systems for us:

```
Mount point: [none] /usr
> w
> q
No label changes.
The root filesystem will be mounted on wd0a.
wd0b will be used for swap space.
Mount point for wd0d (size=66528k)? (or 'none' or 'done') [/var] done
Done - no available disks found.

You have configured the following partitions and mount points:

wd0a /
wd0d /var
wd0e /usr

The next step creates a filesystem on each partition, ERASING existing data.
Are you really sure that you're ready to proceed? [n] y
Warning: 64 sector(s) in last cylinder unallocated
/dev/rwd0a:     132992 sectors in 132 cylinders of 16 tracks, 63 sectors
        64.9MB in 9 cyl groups (16 c/g, 7.88MB/g, 1920 i/g)
/dev/rwd0d:     133056 sectors in 132 cylinders of 16 tracks, 63 sectors
        65.0MB in 9 cyl groups (16 c/g, 7.88MB/g, 1920 i/g)
/dev/rwd0e:     1828512 sectors in 1814 cylinders of 16 tracks, 63 sectors
        892.8MB in 114 cyl groups (16 c/g, 7.88MB/g, 1920 i/g)
-
```

which we will immediately confirm with **done**. Proceed by typing **y** and watch the file systems being created. After this is complete we can configure the network (skip for additional security).

4.2 Configuring The Network

OpenBSD will detect all network cards and give the network devices names that match the driver. For example, our AMD network card gets the name **le1** assigned to it. The network configuration can **easily** be changed, enabled and/or disabled after the installation is complete.

```
System hostname? (short form, e.g. 'foo') openbsd
Configure the network? [y] y
Available interfaces are: le1.
Which one do you wish to initialize? (or 'done') [le1]
Symbolic (host) name for le1? [openbsd]
IP address for le1? (or 'dhcp')          .190
Netmask? [255.255.255.0] 255.248.0.0
Done - no available interfaces found.
DNS domain name? (e.g. 'bar.com') [my.domain]          .edu
DNS nameserver? (IP address or 'none') [none]         0.18
Use the nameserver now? [y] y
Default route? (IP address, 'dhcp' or 'none')      .2.3
add net default: gateway      .2.3
Edit hosts with ed? [n] n
Do you want to do any manual network configuration? [n] n
```

Please note that we have blacked out some of our network configuration, you will obviously have to specify everything correctly.

Next you will have to specify the root password.

4.3 Installing the "install sets"

So far we have nothing but formatted partitions, now we need the actual OpenBSD files.

```
You will now specify the location and names of the install sets you want to
load. You will be able to repeat this step until all of your sets have been
successfully loaded. If you are not sure what sets to install, refer to the
installation notes for details on the contents of each.

Sets can be located on a (m)ounted filesystem; a (c)drom, (d)isk or (t)ape
device; or a (f)tp, (n)fs or (h)ttp server.
Where are the install sets? (or 'done')
```

Assuming that you have the OpenBSD cdrom you can enter **c** here and should see something similar to this – just accept the defaults by pressing enter a few times:

```
duplicate IP address 35.0.220.190 sent from ethernet address 00:09:6b:53:04:02
c
Available CD-ROMs are: cd0.
Which one contains the install media? (or 'done') [cd0]

Pathname to the sets? (or 'done') [3.3/i386]

The following sets are available. Enter a filename, 'all' to select
all the sets, or 'done'. You may de-select a set by prepending a '-'
to its name.

        [X] bsd
        [ ] bsd.rd
        [X] base33.tgz
        [X] etc33.tgz
        [X] misc33.tgz
        [X] comp33.tgz
        [X] man33.tgz
        [X] game33.tgz
        [ ] xbase33.tgz
        [ ] xshare33.tgz
        [ ] xfont33.tgz
        [ ] xserv33.tgz

File name? (or 'done') [bsd.rd] _
```

We want to install everything but the **game33.tgz** so we disable it by entering "**`-game33.tgz`**" and you will see something similar to this:

```
File name? (or 'done') [bsd.rd] -game33.tgz

The following sets are available. Enter a filename, 'all' to select
all the sets, or 'done'. You may de-select a set by prepending a '-'
to its name.

        [X] bsd
        [ ] bsd.rd
        [X] base33.tgz
        [X] etc33.tgz
        [X] misc33.tgz
        [X] comp33.tgz
        [X] man33.tgz
        [ ] game33.tgz
        [ ] xbase33.tgz
        [ ] xshare33.tgz
        [ ] xfont33.tgz
        [ ] xserv33.tgz

File name? (or 'done') [bsd.rd] done
Ready to install sets? [y] y
Getting bsd ...
100% |**************************************************|  4472 KB    00:05
Getting base33.tgz ...
  2% |*                                                 |   692 KB    00:43 ETA_
```

If you are installing onto a 256Mb disk then you will have to skip the **man33.tgz** file as well by entering "**-man33.tgz**" in order to conserve space.

4.4 Finishing Up

Once the install sets have been copied to your disk you will finish up by specifying the timezone, **US/Eastern** for example – but press **?** to see the list. After the installation is complete you will see something like below:

```
Generating initial host.random file...done.
What timezone are you in? ('?' for list) [US/Pacific] US/Eastern
Setting local timezone to 'US/Eastern'...done.
Making all device nodes...done.
Installing boot block...
boot: /mnt/boot
proto: /usr/mdec/biosboot
device: /dev/rwd0c
/usr/mdec/biosboot: entry point 0
proto bootblock size 512
room for 12 filesystem blocks at 0x16f
Will load 7 blocks of size 8192 each.
Using disk geometry of 63 sectors and 64 heads.
 0:  7 @(2 25 57) (9695-9701)
 1: 63 @(2 26 1) (9702-9764)
 2: 26 @(2 27 1) (9765-9790)
 3: 16 @(0 9 47) (613-628)
/mnt/boot: 4 entries total
using MBR partition 3: type 166 (0xa6) offset 63 (0x3f)
...done.

CONGRATULATIONS! Your OpenBSD install has been successfully completed!
To boot the new system, enter halt at the command prompt. Once the
system has halted, reset the machine and boot from the disk.
# _
```

Now enter **halt** and reboot your machine to complete the installation.

5 Post Installation

5.1 Disabling unused services

All standard OpenBSD services are started (configured) in **/etc/rc.conf**, fire up **vi** so we can disable the following services

- inetd
- sendmail

as we won't be needing them. If you intend to only connect to the box locally (through a keyboard or serial console) then I recommend disabling the **sshd** also. The lines should look like this after our little modification:

```
sendmail_flags=NO
inetd=NO
```

and possibly

```
sshd_flags=NO
```

These changes will be in effect after a reboot. Remember, to activate a previously disabled service simply change the **NO** to **""** unless you want to pass additional flags to the application.

5.2 Removing unused directories

This step is absolutely optional but cleans up the **/var** directory a bit. Since we won't be using a httpd, named and yp we can remove some of the directories in **/var**. If you intend to use any of these services make sure you **do not** remove the directories. Now type

- `rm –rf /var/www`
- `rm –rf /var/games`
- `rm –rf /var/named`
- `rm –rf /var/yp`

5.3 Mounting /usr read-only

Since we are trying to build a system that is mostly read-only we will start with the easiest step, mounting **/usr** read-only. Open up **/etc/fstab** and make sure the line mounting **/usr** looks like this:

```
/dev/wd0e /usr ffs ro,nodev 1 2
```

We essentially changed **rw** to **ro**. We'll be adding more to this file later.

5.4 Installing additional packages

We will utilize **rsync** to copy files from and to our ramdisk, since it is not included we will have to manually add it. Type the following commands:

- `mkdir /cdrom`
- `mount /dev/cd0a /cdrom`
- `pkg_add /cdrom/3.3/packages/i386/`**`rsync-2.5.6.tgz`**

We personally prefer the bash shell over other shells so we'll install that as well by typing

- `pkg_add /cdrom/3.3/packages/i386/`**`bash-2.05b-static.tgz`**

Note that both of these applications will be installed in the `/usr/local` directory.

5.5 Removing other unused features

Cron has quite a few jobs running in regards to sendmail which we will disable along with disabling the backup of the configuration files in /etc. Comment out line 9 so that it looks like this:

```
#*/30   *   *   *   *    /usr/sbin/sendmail –L sm-msp-queue –Ac -q
```

To disable backup of the configuration files to /var add

ROOTBACKUP=0

under **HOME=/var/log** (line 5).

5.6 Enabling a serial console

In order to log in to our server through the serial port we will need to activate this in the file **/etc/ttys**. Change the following line

```
tty00 "/usr/libexec/getty std.9600" unknown off
```

to the following:

```
tty00 "/usr/libexec/getty std.9600" vt100 on secure
```

Before you reboot the machine note two important things. First the word **secure** means that root can logon on **tty00** which is otherwise disabled. If you don't need to logon as root on that console simply skip the **secure**.

If you are having trouble logging on (after a reboot) from your other PC (e.g. Windows using Hyperterminal) then you might want to try and change the leading **tty00** to **cua00**.

If you do not plan on attaching a monitor to this box at all then you might want to create a file **/etc/boot.conf** with the following content:

```
set tty com0
```

in order to send boot messages to the serial console rather than showing them on the screen.

5.7 Enabling ramdisks

OpenBSD has a file system called **mfs** which stands for memory file system. It allows you to create an entire file system in memory (RAM). Since CompactFlash disk have a limited amount of write cycles we will use this feature to our advantage. We will create the following directories in mfs:

- `/tmp` 32Mb
- `/var/log` 64Mb
- `/var/run` 1Mb
- `/var/tmp` 8Mb

by adding the following lines to **`/etc/fstab`**:

```
swap /tmp mfs rw,-s=65536,nodev,noatime,noexec,nosuid 0 0
swap /var/log mfs rw,-s=131072,nodev,noatime,noexec,nosuid 0 0
swap /var/run mfs rw,-s=2048,nodev,noatime,noexec,nosuid 0 0
swap /var/tmp mfs rw,-s=16384,nodev,noatime,noexec,nosuid 0 0
```

As you have no doubt noticed the **`-s`** switch in the mount options specifies the actual size of the mfs partition in 512 byte blocks. Hence **`65536`** are actually 32 megabytes.

We specify a few additional security and performance options:

nodev no device files will be interpreted
noatime the access times of the files won't be updated
noexec no executable files on this file system
nosuid Ignore set-user-identifier and set-group-identifier

These partitions will be mounted on top of existing directories and the original content will not be accessible unless the mfs file systems are umounted.

5.8 Read-Only 1: Automatically copying...

MFS file systems are nice and fast but they do not retain their data through a system reboot. For this reason we will save all files from a few of these directories on a special directory that we will create: **`/mfs/var.log`**. So type **`mkdir –p /mfs/var.log`** to create this directory.

Upon system shutdown we will copy all files from the **`/var/log`** directory to the **`/mfs/var.log`** directory. This can be automated from the **`/etc/rc.shutdown`** file.

When the system boots we will fill the **`/var/log`** directory from the **`/mfs/var.log`** directory by adding a line to the **`/etc/rc`** file – using rsync. Type the following lines to create an initial copy of the **`/var/log`** directory. The first line merely gets rid of any compressed log files that might have been created since you installed OpenBSD:

```
rm /var/log/*.gz
/usr/local/bin/rsync -vorpug /var/log/ /mfs/var.log
```

and add the following to the **`/etc/rc`** file just after the "**`mount /var >/dev/null 2>&1`**" line:

```
# Fill /var/log directory
printf "copying files to mfs …"
/usr/local/bin/rsync –orpug /mfs/var.log/ /var/log
echo " done."
```

and reboot the machine. After a reboot the contents of the **`/var/log`** directory are located in memory.

Now we want to synchronize the content of the **/var/log** directory when we reboot our machine in the future. Append the following lines to **/etc/rc.shutdown**

```
# Save /var/log directory
printf "Copying /var/log to CF …"
# ReMount / read-write now
mount –uw /dev/wd0a /
/usr/local/bin/rsync –orpug --delete /var/log/ /mfs/var.log
echo " done."
```

and reboot the machine one more time to see it in action. The "mount –uw …" line is necessary since we are going to mount the root file system read-only later.

5.9 Read-Only 2: Creating /dev in mfs

In order to achieve a truly read-only installation we will need to also create the device files in mfs. Type the following commands:

```
mkdir /mfs/dev
cp /dev/MAKEDEV /mfs/dev
cd /mfs/dev
./MAKEDEV all
```

to create all device files in the `/mfs/dev` directory. Now we can mount the `/dev` directory on top of `/dev` in mfs as well. Add the following line to **/etc/fstab**

```
swap /dev mfs rw,-s=1024,nosuid 0 0
```

and since we're already at it change the first three lines to

```
/dev/wd0a / ffs rw,noatime,softdep 1 1
/dev/wd0e /usr ffs ro,nodev,noatime,softdep 1 2
/dev/wd0d /var ffs rw,nodev,nosuid,noatime,softdep 1 2
```

where changes are shown in bold. And then add the following to line

```
# Copy dev files before anything else
cp –Rp /mfs/dev/* /dev
# ReMount / read-only
mount –ur /dev/wd0a /
```

to **/etc/rc** after the line **rm –f /fastboot**

By default OpenBSD remounts / read-write (regardless of the **ro** setting in **/etc/fstab**) which we can prevent by commenting out the line

```
mount –uw /
```

just above the line with **rm –f /fastboot**.

5.10 Summary

We modified a whole bunch of files that you might want to save for future reference and/or sanity. Here are all the files that we modified in no particular order:

1. /etc/fstab CF only
2. /etc/rc.conf
3. /etc/rc CF only
4. /etc/rc.shutdown CF only
5. crontab –e CF only
6. /etc/ttys
7. (/etc/boot.conf)

In addition we created the following directories

1. /mfs/dev (incl. MAKEDEV and all devices) CF only
2. /mfs/var.log (incl. log files) CF only

and we deleted the following directories

1. /var/www CF only
2. /var/games CF only
3. /var/named CF only
4. /var/yp CF only

and installed the following packages

1. bash-2.05b-static
2. rsync-2.5.6 CF only

At this point we should have a working OpenBSD installation and we are ready to configure all of our networking options now.

Important Note: In order to configure our system further we will need write access to our root partition, something we just disabled before. To remount the root file system read-write enter the following command:

```
mount –uw /dev/wd0a /
```

and enter this command to remount root read-only after you are done

```
mount –ur /dev/wd0a /
```

6 Networking

We will now configure our OpenBSD box to act as an ethernet bridge and setup PF (packet filtering) so that we can filter out evil packets.

6.1 Configuring the network devices

We will use the network interfaces **`rl0`** and **`rl2`** as our bridging interfaces, **`rl0`** being the internal and **`rl2`** being the external interface. To tell OpenBSD that we want these interfaces up yet do not want an IP address we create two files:

- `/etc/hostname.rl0`
- `/etc/hostname.rl2`

that contain only one line:

```
up
```

This is equivalent to typing **`ifconfig rl0`** up. We will however configure interface **`rl1`** to be our management interface that we can connect to in order to perform administrative tasks.

6.2 Configuring a management network device

Create the file **`/etc/hostname.rl1`** with the following content:

```
inet 10.10.10.1 255.255.255.0 NONE
```

To activate this interface immediately you would have to type **`ifconfig rl1 10.10.10.1 netmask 255.255.255.0`**, otherwise you will have to reboot the machine or run **`sh /etc/netstart`**. You can obviously choose a different IP address if you would like, just make sure you choose one that is from a private IP range so it is not accessible from the Internet.

Usually you would now also configure the files **`/etc/mygate`**, **`/etc/hosts`** and **`/etc/resolv.conf`** but we will refrain from editing these now.

6.3 Creating the bridge

To start up the bridge immediately type the following command:

```
brconfig bridge0 add rl0 add rl2 up
```

and to permanently create this bridge create the file **`/etc/bridgename.bridge0`** with the following content:

```
add rl0
add rl2
up
```

This will create bridge **bridge0** when the system boots up.

6.4 Activating the packet filter

To activate the packet filter immediately you can simply type **`pfctl –e`**, to always activate PF open the **`file /etc/rc.conf`** and change the line

```
pf=NO
```

to

```
pf=YES
```

6.5 Filtering packets

After issuing the **`pfctl –e`** command or rebooting the system your OpenBSD box will begin forward ethernet frames from one NIC (**`rl0`**) to the other (**`rl2`**) and also filter packets according to the rules found in **`/etc/pf.conf`**. The rules in this file are, as is to be expected, very open allowing all traffic to pass through.

Unfortunately it is outside the scope of this guide to give you a detailed explanation on how packet filtering, TCP/IP and such work. Hence we assume that you have a basic understanding of the OSI

model, TCP/IP and internet applications.

We will create a simple ruleset later that will disallow the following:

- incoming **SSH** traffic from the outside is blocked (with one exception)
- incoming **NetBIOS** traffic to protect our Windows boxes
- outgoing traffic to file sharing services

and **allow everything else**. Please note that PF rules are read sequentially and rules appearing last will take priority previously matched rules.

Usually firewall rulesets will block everything by default and only allow certain services, however we will take a different approach and only block services known to cause trouble. Feel free to change our default rules to block everything by default instead.

6.6 Reloading rules

Changing the pf.conf file does not automatically activate them, this needs to be done with the **`pfctl`** command. To reload the rules type

```
pfctl –f /etc/pf.conf
```

If you want to flush all rules and hence permit all traffic to go through you can type

```
pfctl –F all
```

6.7 Testing the rules

To test whether a particular ruleset has the correct syntax type

```
pfctl –n –f /etc/pf.conf
```

7 Example PF Rules

This chapter will explain the style of the pf.conf file and also contain a few real-world examples and how you can implement them using PF. The actual rules that you will add to this file will of course depend on your environment.

7.1 Definitions

To ease maintenance of the **`pf.conf`** file we can define certain static information, such as the interface names, network IP ranges and more at the beginning of the file. Note that these definitions have to appear before the actual rules.

```
# Interface definitions
ext_if="rl2"
int_if="rl0"

# Internal network
internal_net="10.10.10.0/24"
```

7.2 Defining the default mode

Since our firewall will allow all traffic we add the following line **after** the definitions and **before** the rules:

```
pass in all
pass out all
```

7.3 Blocking services using a single port

To block incoming SSH traffic we can use the following rule

```
block in on $ext_if proto tcp from any to $internal_net port 22
```

7.4 Blocking services using multiple ports

Some services, such as NetBIOS, use more than one port. In this case it might be easier to define the range of ports first and then refer to this range in the actual rule. The port range for NetBIOS (incl. LDAP and Terminal Services) is defined by adding the following line to the top of the file, before the rules:

```
ports_win = "{ 137, 138, 139, 445, 389, 3389 }"
```

and the actual rule is

```
block in on $ext_if proto { tcp,udp } from any to $internal_net port $ports_win
```

7.5 Blocking outgoing file sharing protocals

We can block outgoing connections (TCP & UDP) for the following file sharing applications:

```
1044    Direct File Express
1045    Direct File Express
1214    Kazaa, Grokster
4329    iMesh
4661    Edonkey 2000
4662    "
4665    "
6346    BearShare, Gnucleus, Morpheus, Swapper, XoloX, LimeWire,
        Phex, Gnewtellium, Gtk-Gnutella, Mutella, Qtella
6347    "
6666    Yoink
6667    Yoink
6699    AutoNap, BeNapster, Napster, Crapster, Duskter, Gnap,
        Gtk Napster, Hackster, iNapster, Jnap, WinMX
6700    "
6701    "
7668    Aimster / Madster
7788    BuddyShare
8311    Scour
8888    AudioGnome, OpenNap, Swaptor
8889    AudioGnome, OpenNap
28864   hotComm
28865   hotComm
41170   Blubster
```

We will block access to these services by adding the following lines to the **beginning** of the **`pf.conf`** file:

```
ports_ptp = "{ 1044, 1045, 1214, 4329, 4661, 4662, 4665, 6346, 6347, 6666, 6667,
6699, 6700, 6701, 7668, 7788, 8311, 8888, 8889, 28864, 28865, 41170 }"

block out on $ext_if proto { tcp, udp } from $internal_net to any port $ports_ptp
```

It is **important to note** that some file sharing applications/protocols, most notable Kazaa, will dynamically switch and use different ports (including HTTP 80) when they are unable to talk to the servers. The rules above may therefore only prevent certain less sophisticated protocols. For more information on how to prevent Kazaa please search the **Misc OpenBSD** mailing list, **Google Groups** or wait for an updated version of this document.

7.6 Exceptions to the rule

Where there are rules there are exceptions. The rules contained in the pf.conf file are prcoessed sequentially, and later matching rules have priority over earlier matching rules. Hence it is wise to declare the more general rules (blocking in our example) in the beginning and the more specific rules after.

For example, we blocked all **SSH** traffic earlier on but can make an exception for the host with IP address **1.2.3.4**. Simply add this line after the previous (see 5.3.) block rule:

```
pass in on $ext_if proto tcp from any to 1.2.3.4 port 22
```

7.7 Logging dropped packets

If you would like to log dropped packets rather than simply ignoring them you can add the **log** statement to a rule. For example, to block all incoming SMTP traffic and log all attempts to connect to this service add:

```
block in log on $ext_if proto tcp from any to $internal_net port 25
```

8 Appendix A: Installation on a DELL Optiplex GX270

The default OpenBSD 3.3 kernel has some issues on a DELL Optiplex GX270 that prevent the OS from booting. This seems to be related to a S-ATA controller that cannot be disabled in the BIOS. Please follow the following steps to install OpenBSD on DELL Optiplex GX270, assuming you make no use of the S-ATA controller:

1. Booting from the installation CD
Boot from the OpenBSD CD, wait for the **boot>** prompt and enter

```
boot> boot –c
```

to modify the running kernel. After you get the **UKC>** prompt type

```
UKC> disable 50
UKC> quit
```

which disables the pciide device. In case you are curious, this number is obtained by typing **find pciide** which would yield output similar to this:

```
 50 pciide* at pci* dev –1 function –1 flags 0x0
```

Then install OpenBSD as described earlier. Once OpenBSD is installed reboot and wait for the **boot>** prompt again and enter

```
boot> boot –c
```

one more time. At the **UKC>** prompt then enter

```
UKC> disable 75
UKC> quit
```

and boot the OS. The number is **75** now since the kernel installed by OpenBSD is different to the kernel included with the CD. After you have logged in enter the following commands on the shell to write the changes to the kernel (`/bsd`) stored on the disk:

```
# config –e –f –u /bsd
ukc> quit
```

9 Appendix B: Example of pf.conf file

```
# /etc/pf.conf
# Example
#

# Interface definitions
ext_if="rl2"
int_if="rl0"

# Internal network
internal_net="10.10.10.0/24"

# Popular Ports
ports_win = "{ 137, 138, 139, 445, 389, 3389 }"

ports_ptp = "{ 1044, 1045, 1214, 4329, 4661, 4662, 4665, 6346, 6347, 6666, 6667,
6699, 6700, 6701, 7668, 7788, 8311, 8888, 8889, 28864, 28865, 41170 }"

# Default Rules
pass in all
pass out all

# Block & Log incoming SSH
block in log on $ext_if proto tcp from any to $internal_net port 22

# Block windows
block in on $ext_if proto { tcp,udp } from any to $internal_net port $ports_win

# Block some outgoing peer-to-peer
block out on $ext_if proto { tcp, udp } from $internal_net to any port $ports_ptp

# Allow this SSH connection
pass in on $ext_if proto tcp from any to 1.2.3.4 port 22
```

10 Appendix C: Additional Resources

If you need more information on OpenBSD then we recommend the following sources.

Learning more about OpenBSD
The OpenBSD FAQ is the official OpenBSD documentation and contains a lot of information. It is very well written and easy to read. This should be your first step in your quest for more information.

Absolute OpenBSD
If the FAQ does not contain enough information then you might want to consider getting this book, maybe from Amazon. It is very well written and understandable and also covers a lot about OpenBSD. As of writing this is the only "general" OpenBSD book available. For more books check out http://www.openbsd.org/books.html.

www.openbsd.org
No doubt the best resource for OpenBSD is the OpenBSD website http://www.openbsd.org. It's very well structured and points you in all the right directions.

If you have a specific problem about OpenBSD you should sequentially check the following resources:

1. Previously mentioned resources
2. Man Pages (these are very will written)
3. Google Groups

4. OpenBSD Mailing List Archive (e.g. misc@openbsd.org)

If you can't solve the problem after checking these resources then you can join the misc@openbsd.org mailing list and send an email there. People are generally **very** helpful on this list. Read this first before you post a message though.