How to Build a FreeBSD-STABLE Firewall with IPFILTER

Applicable to: FreeBSD 4.6
Updated: Sep 3, 2002
Author: Marty Schlacter
Source URL:
http://www.schlacter.net/public/FreeBSD-STABLE_and_IPFILTER.html

This howto walks you through the process of building one of the most stable and secure firewalls available - a FreeBSD-STABLE firewall with IPFILTER. As a part of the installation process, all services will be disabled except OpenSSH, which will have its access controlled via TCP-Wrappers. The firewall will be configured to log through the syslog facility, but will have its own firewall log files (rather than filling up /var/log/messages). We'll add VESA support into the kernel so that we can use 132x43 screen resolutions, as well as compile support into the kernel for a second ISA Ethernet card if you have one. After we add a warning banner to the system, we'll make BASH the default shell for root, perform a rudimentary setup for root's BASH environment, and redirect root's email to your "normal" account so that the root account on the firewall itself doesn't fill up. Next, we'll download, compile, install, and configure Tripwire, as well as install cvsup so that your ports collection stays up to date. And, lastly, we'll modify the /etc/fstab entries so that some of your partitions are mounted 'nosuid', 'noexec', or 'ro' so that your installation is as secure as possible.

This is an all-encompassing how-to, and should take most of a complete day to complete, but when you're finished, you'll not only have a great firewall, but will be better able to compare and contrast FreeBSD/IPFILTER to Linux/IPTABLES or OpenBSD/PF so that you can consider the pros/cons of each on their merits...and that learning process is what all of this about anyway. So, grab a cup of coffee, sit down with that old Pentium, and get ready to broaden your horizons.

Before we start, I'd like to thank Dan O'Connor for the work he put in on his great site, FreeBSD Cheat Sheets, since it was his great site that gave me the motivation to start this howto. You will undoubtedly see some of his tips and tricks sprinkled throughout this document. For those of you that are new to FreeBSD, I highly recommend his site. His site is a little out of date, due to changing priorities in his life, but the info on his site is still very applicable to any version of FreeBSD 4.X.

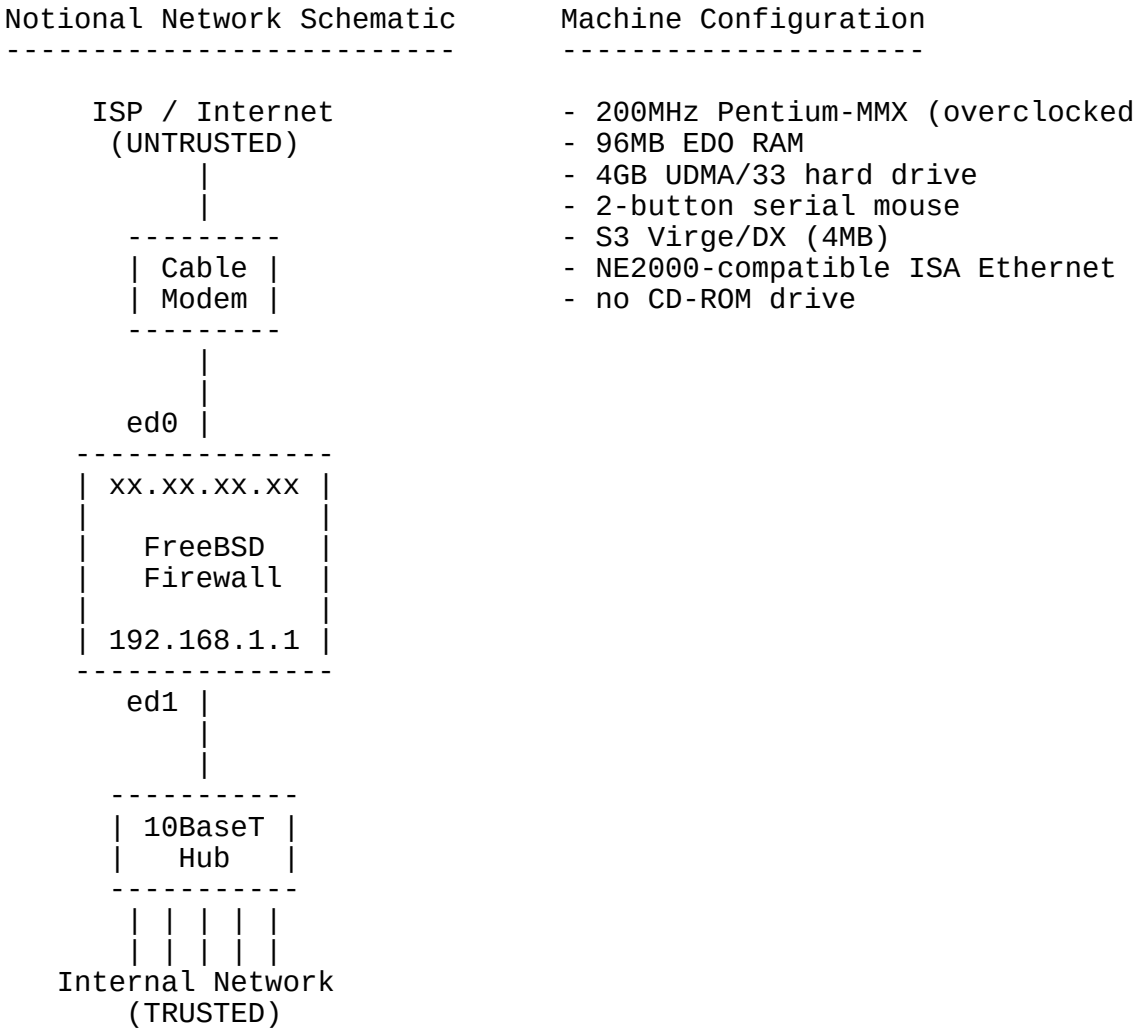
In addition, there have been several other people on the Internet who have given me great suggestions & and feedback on this HOWTO. The majority (if not all) of their comments have been incporporated into this document in some form or another. There are too many to list here by name, but (rest assured) the Open Source community has helped to make this the best document it can be.

And, as always, before performing this procedure, I highly recommend that you review the Installing FreeBSD chapter of the FreeBSD Handbook.

Network Schematic & System Configuration

The intent of this document is to show you how to build a firewall for your home

network. Just to make sure that we're "working off the same sheet of music" here's a quick ASCII-schematic of what our notional home network will look like - to include device names for the Ethernet interfaces. In addition, I'm including a quick synopsis of the configuration of my own hardware - so that you can use it as a reference point throughout this procedure.



Installing FreeBSD

To build the most stable and security-patched system you can, you'll want to make sure you're running the latest version of FreeBSD-STABLE. For those of you new to FreeBSD, the STABLE branch is the version of the operating system that has all of the latest patches, bugfixes, and enhancements after the previous release was made. In fact, there's actually two different versions of the STABLE branch...one that has all of the patches, bugfixes, and enhancements, and a second that only has the bugfixes and patches (no enhancements). The second version is usually more stable than the first, but not always so. For a production firewall, you'll probably want to install the 2nd version of STABLE (without the enhancements), but it's ultimately your call.

If you've installed FreeBSD-4.6 from CD-ROM (either one that your purchased or 'burned' from a downloaded ISO image), you probably installed 4.6-RELEASE, which is (simplistically) nothing more than a version of the 4.X branch that was exhaustively tested, burned to CD-ROM and made available for sale. After the

release date of 4.6-RELEASE, the 4.6 tree continued to evolve & be patched (for security reasons) after that point. Since there's no way the folks at FreeBSD.org can burn & sell CD-ROMs for each day's version of the 4.6 tree, 4.6-RELEASE is the only one made available for sale on CD, and subsequent snapshots of the 4.6 tree are only available on-line and are labelled '4.6-STABLE'. Once 4.6-STABLE is sufficiently enhanced/patched (perhaps 4-6 months later), the code enters a freeze and will officially become the 'RELEASE' version of the next FreeBSD release (say, 4.7-RELEASE...or 5.0-RELEASE). If you're installing FreeBSD 4.6 well after the release date, you will definitely want to install 4.6-RELEASE, and then immediately update your kernel and binaries to 4.6-STABLE.

So, what are the benefits of upgrading to 4.6-STABLE rather than staying with 4.6-RELEASE? Well, the biggest answer (if you're building a firewall, like we are here) is that all of the security patches have been applied to the O/S and the associated applications. To use a prior baseline of FreeBSD (4.2) as an example, FreeBSD-4.2-RELEASE (which was released in November 2000) uses OpenSSH-2.2.0, which is a great product but also has a remote buffer overflow that wasn't discovered until early February, 2001. If a hacker exploited this vulnerability on your 4.2-RELEASE box, they would gain remote root access and ruin your day. The relevant info on this vulnerability can be found on SecurityFocus' website. When you upgraded to FreeBSD-4.2-STABLE (if you were following this HOWTO in mid-March of 2001), by comparison, you would have gotten FreeBSD-4.2-RELEASE with all of the patches applied after the November 2000 release...so your system would have OpenSSH-2.3.0 (not OpenSSH-2.2.0) which is not vulnerable to the remote buffer overflow. So upgrading to the latest snapshot from the STABLE branch saves you a lot of time associated with loading individual security-related patches after your OS load is finished. For a complete listing of security-related patches, see the FreeBSD Security Information page.

OK, now that we've talked about the benefits of FreeBSD-STABLE, let's get to work...the installation...

1. Inventory your computer hardware and ensure that it is compatible with FreeBSD. The latest compatibility list (for the 4.6 baseline) can be found in the FreeBSD 4.6 Hardware Notes.

2. Verify that you have at least 1.1G available on your hard drive. After the initial install of FreeBSD (the first section of this document), you will have taken up about 350M. After downloading the latest kernel sources, and updating your ports tree, you will have taken up about 650M (depending on the number of ports sections you wish to keep up to date). And, finally, after you finish installing & compiling tripwire and recompiling the kernel, you will have taken up about 1.1G. Which directories are the biggest disk space hogs? /usr/obj (& sub-directories) takes up about 377MB. /usr/src (& sub-directories) takes up about 350MB. /usr/ports (& sub-directores) takes up about 160MB. All other directories take up less than 90MB apiece.

3. Download the boot floppy images:

    A. FTP to ftp://ftp.freebsd.org/

    B. Change directory into /pub/FreeBSD/releases/i386/4.6-RELEASE/floppies

C. Download the kern.flp and mfsroot.flp images & store them in your /tmp directory (on Linux or FreeBSD) or c:\windows\temp directory (for Windows), depending on what system you're downloading from.

260 Download the floppy creation tools if you're a DOS/Windows users.

A. FTP to ftp://ftp.freebsd.org/

B. Change directory into /pub/FreeBSD/tools

C. Download the program, fdimage.exe, and store it in the same directory that you used, above.

261 Create Boot Floppies

A. If you're using Linux or FreeBSD, use the dd command as follows, and create one floppy from the kern.flp image, and another disk from the mfsroot.flp image.

```
[root@yoursys /tmp]# dd if=/tmp/kern.flp of=/dev/fd0
2880+1 records in
2880+0 records out
1474560 bytes transferred in 49.931306 secs (30135 bytes/sec)
```

B. If you're using DOS/Windows, use the fdimage program that you downloaded. Just like with Linux, make one floppy from the kern.flp image, and another one from the mfsroot.flp image.

```
C:\WINDOWS\TEMP>fdimage kern.flp A:
C:\WINDOWS\TEMP>fdimage mfsroot.flp A:
```

262 On the FreeBSD machine, insert the kernel floppy (kern.flp) in your floppy drive and boot from it. When prompted, insert the 'MFS root' floppy (mfsroot.flp).

263 Run the kernel configuration utility in full-screen visual mode to clear any conflicts and ensure the kernel matches your hardware. For example, remove SCSI controllers if you don't have any, etc. On my system (where I don't have any SCSI controllers or a PS/2 mouse), here's the only active drivers I left enabled (I deleted the rest):

```
Storage:
   ATA/ATAPI compatible disk controller          ata0   14 0x1f0
   ATA/ATAPI compatible disk controller          ata1   15 0x170
   Floppy disk controller                        fdc0   6  0x3f0
Networks:
   NE1000,NE2000,3C503,WD/SMC80xx Ethernet
adapters                                         ed0    10 0x280
Communications:
   Parallel Port chipset                         ppc0   7
   8250/16450/16550 Serial port                  sio0   4  0x3f8
   8250/16450/16550 Serial port                  sio1   3  0x2f8
Input:
```

```
   Keyboard                                    atkbd0 1
   Syscons console driver                      sc0
Multimedia:
Miscellaneous:
   Math coprocessor                            npx0   13 0xf0
```

Note: If you have PCI-based Ethernet cards, you can delete all of the network cards in the list - yours will be found and configured automatically. If you're on the other end of the scale (like me) and you have two old NE2000-compliant ISA network cards, you'll only be able to configure one of them at this time (ed0). After your installation is complete, you'll have to build a custom kernel & add in a "placeholder" for the 2nd generic ISA card, and then run through the kernel configuration utility again after you reboot. We'll do this at the end of this document.

Hit **'Q'** then **'Y'** to save your changes and exit.

380. From the main menu, choose a **'Standard'** installation.

381. In the *FDISK Partition Editor*, first **'D'** delete any disk slices that already exist, then choose **'A'** to use the entire disk. This will let FreeBSD take the entire disk and eliminate the need for a bootloader. Press **'Q'** to continue.

382. Now, you will now be presented with the *Install Boot Manager for drive...* screen. Select **'Standard'** to install a standard MBR (no boot manager). After all, you won't be dual-booting this machine...it's your firewall. Therefore, you won't need a boot loader.

383. In the *Disklabel Editor*, create the following partitions, then choose **'Q'** to continue. Note that I'm using a 4GB hard drive. You can decrease the sizes of the partitions if you don't have a 4GB hard drive for your system. The /usr/local and /usr/home partitions can go as low as 64MB since this won't be a common-user system and there won't be a lot of user-specific files or binaries...but the /usr partition should never go below 650MB since that's where all of your kernel source code and ports tree is located. Here's a partition scheme if you have a 4GB drive:

```
256MB swap partition (or at least 2x your RAM)
128MB file system mounted as /
512MB file system mounted as /tmp
512MB file system mounted as /var
1,500MB file system mounted as /usr
640MB file system mounted as /usr/local
500MB file system mounted as /usr/home (...the remainder of the
hard drive)
```

Here's a partition scheme if you only have one of those old 1.1 GB drives. People have reported success when using this partitioning scheme on a drive this small. But, as always, 'caveat emptor'. You'll probably run out of space if you're not careful. One recommendation is to not install the ports collection at all. That'll save about 160MB in the /usr partition. Another recommendation is to only re-compile the kernel and not all of the system binaries (i.e. only run the "build kernel" command when you get to the appropriate section at the end of this howto). Apply security-related patches

to the system binaries manually by following the directions for each patch listed on the FreeBSD Security Information page. Yes, it's a pain...but if your hard drive is too small, then it's too small...

```
128MB swap partition
128MB file system mounted as /
64MB file system mounted as /tmp
64MB file system mounted as /var
640MB file system mounted as /usr
64MB file system mounted as /usr/local
32MB file system mounted as /usr/home
```

424. Choose "Kern-Developer" as the Distribution you want to install by highlighting it and pressing the **'space'** bar. Remember, this is going to become a gateway/firewall system, and you'll need the kernel source code to recompile IPFILTER into the kernel. Also, you don't need (or want) X Windows running on it.

425. Select "Yes" to install the FreeBSD ports collection.

426. Arrow back up to "<<< X Exit" and hit the **'space'** bar to exit the Distribution Menu

427. Select either an FTP or FTP Passive install (depending on what your current network's firewall will support).

428. Either select the "Primary Site (ftp.freebsd.org)" or select a FTP site in your country nearest your location. Arrow down to your selection and press [Enter].

429. Select your Ethernet card as the network interface to install from (e.g. "ed0" if you're using a generic NE2000-compatible ISA card).

430. Select "no" for IPv6 config

431. Select "yes" for DHCP configuration if your network card is directly connected to your cable modem, etc. Select "no" if you're on a pre-existing network, then enter your interface configuration information manually - host name, domain name, IPv4 gateway IP address, name server IP address, IPv4 address, and netmask.

432. At the "Last Chance" warning, select "yes".

(System Installs...If the FTP site that you chose isn't heavily loaded, the install can take as little as 22 minutes (with a cable modem). If your FTP site is heavily loaded, the install can take as long as 2 hours...or longer...)

433. Miscellaneous configuration:

A. Do you want this machine to function as a gateway? Yes

B. Do you want to configure inetd and simple internet services? No

C. Do you want to have anonymous FTP access to this machine? No

D. Do you want to configure this machine as an NFS Server: No

E. Do you want to configure this machine as an NFS Client: No

F. Select "No" when asked "Do you want to select a default security profile for this host". This will select the "Medium" setting. We will change this to the "Extreme - Very restrictive security settings" at the end of this procedure - after we recompile the kernel, etc.

G. Select "No" when asked to modify the system console configuration.

H. Select "Yes" when asked "Would you like to set this machine's time zone now?" Then, select "No" when asked if your machine's CMOS clock is set to UTC. Then select the appropriate time zone - by region, country, and then the applicable time zone.

I. Select "No" when asked if you'd like to install Linux Binary support.

J. Select "Yes" when asked if your system has a non-USB mouse attached to it (unless, of course, you don't...)

K. Make the following configuration changes for the mouse configuration, then enable it & test it, then select "Exit" to return to the previous menu. Note that I have a 2-button serial mouse - that's why I'm using COM1 and 3-button emulation:

```
Type: Auto
Port: COM1
Flags: -3
```

L. When asked to browse the FreeBSD packages collection, select "Yes", and then install the following packages. Note that these package preferences are just my own personal preferences. If you're a firewall 'purist' (which means you take a more minimalistic approach when configuring firewalls - for security reasons) then the only package you'll need to install is cvsup (so that you can get the latest copy of the source & ports, etc.) If you're like me, I like using lynx to access the web, mutt to read email, and bash as my shell. Even though I don't use the firewall as a common-user machine, I consider those three programs "necessities" for me. Your usage patterns will vary. Regardless of what my own preferences are, please substitute, add, or delete as you see fit...it's your firewall after all...

```
WWW - lynx-ssl-2.8.4.1b_1
Mail - mutt-1.2.5.1_1
Net - cvsup-without-gui-16.1f
Shells - bash-2.0.5a
FTP - ncftp-3.1.2
```

Then tab over and select "Install", select "OK" to confirm your choices

(Packages are installed...takes about 60 seconds)

M. Select "Yes" when asked if you want to add any additional user accounts. Since this is a firewall, not a common user machine, we

won't need many, but you will need at least one. The main reason we're adding at least one other user account is so that we can set up SSH so that it does not allow remote root logins. Instead, you must SSH to the firewall as the user, and then 'su' to root.

N. Select "User - Add a new user to the system" on the *User and group management* dialog box. Then enter the login id, password, and full name. Make sure you put a '0' in the member groups box. This will put your new user in the 'wheel' group so that they can 'su' to root. Also put /usr/local/bin/bash in for their default shell. When finished, select 'OK', and then 'X - Exit'

O. Set the 'root' password: ******

P. When asked if you'd like to visit the General Configuration menu to set any last options, select "Yes" and configure the following options:

```
Networking:
  - Disable "inetd - This machine wants to run the inet
daemon" ... then select "No" to confirm
  - Enable "ntpdate - Select a clock-synchronization server"
... then select a server near you
```

Then select Exit and return to the previous menu, and then tab over and select "Exit Install"

Q. Select OK when asked if you're sure you want to exit the install & reboot the system. Remove your floppy disk (probably the mfsroot disk) and your system will reboot.

(System reboots...)

Upgrading to -STABLE, Compiling IPFILTER into the kernel, & Configuring the System

Now that you have FreeBSD-RELEASE installed on the system, we need to spend a few hours upgrading to FreeBSD-STABLE, adding in IPFILTER support, as well as finishing the rest of the configuration. Here's what we're going to do in this section (in no particular order):

- Configure cvsup and update your source tree & ports collection
- Upgrade to FreeBSD-STABLE
- Compile IPFILTER into the kernel and configure IPFILTER, IPNAT, and IPMON
- Configure IPMON so that it logs to syslog, but modify syslog so that the firewall messages get their own file and then update newsyslog so that the firewall's logs get rotated
- Install and configure Tripwire
- Compile VESA support into the kernel and change our screen resolution is 132x43
- Configure syslogd so that it won't accept connections from other machines (i.e. prevent it from being a 'listening' service)
- Add support for (and configure) a 2nd Ethernet interface (if you have 2 ISA cards)
- Configure TCP-Wrappers so that access to SSH is locked down to your local

network only
- Configure SSH so that it will only accept SSH sessions from IPv4 systems and rejects connections from users it doesn't have the DSA key for
- Disable unused services in inetd (just in case it accidentally gets turned on later)
- Add a warning banner
- Make BASH the default shell for 'root' & configure root's BASH environment
- Redirect root's email to your "normal" account so that it doesn't back up on the firewall.
- Modify the /etc/fstab so that some of the partitions are mounted 'nosuid', 'noexec', or 'ro' to lock the system down even further.
- Increase the kernel's security level to "2" (Extreme)

In order to save time, I'm going to do some steps in what will appear to be an "out of order" sequence. This is being done on purpose so that we will minimize the number of re-boots you'll have to do. In fact, the goal is to configure the system, then recompile the kernel & system binaries, and when the system reboots, you're done. That's it.

1. Log in as your non-priveleged user account. If your login was successful, you should be presented with a 'bash-2.05a$' prompt...indicating that bash was successfully installed. After you log in, then type 'su' to switch user to root. Enter the root password.

2. Make "bash" the default shell for 'root' and perform an initial set up of root's bash environment.

    A. Use FreeBSD's password file manipulation utility, vipw, to modify root's default shell. At a root prompt, type vipw. A copy of the /etc/passwd file will be displayed. Use standard vi editing commands to change root's default shell from /bin/csh (all of the way at the end of the first line) to /usr/local/bin/bash. While you're already editing the file, go ahead and change root's unofficial name 'Charlie &' to 'Super-User' or any other name that envisions Superman, etc. When you get mail from root (e.g. from the cron jobs that run every night), it'll now be maked as coming from 'Super-User' and not 'Charlie &'...just a little bit nicer. Save & exit.

    B. Verify that your manipulation of the password file was successful. Go over to your 2nd virtual terminal by hitting <Alt>-F2. When you're at the 2nd virtual terminal, log in as root. After successfully logged in, verify that you're presented with the 'bash-2.05#' prompt. If it's successful, then log out and return to the 1st virtual terminal to continue working. If it's not successful, then you need to go back to the previous step and figure out what you did wrong. Remember that bash is working because you logged in as your user account. You must have typed in something wrong, or accidentally removed a ':' (colon), etc. Go back to the first virtual terminal, type 'vipw' and re-edit the password file to fix your mistake.

    C. Create a .bashrc file in root's home directory (/root) and enter the following items (as a starting point). After the file has been created, chmod 600 on it so that it's only readable & writable by root. Then

copy it to your user's home directory (cp /root/.bashrc /usr/home/username/.bashrc). And, lastly, do a chown on the file in your user's directory so that they own the file (not root), by doing a 'chown username:groupname /usr/home/username/.bashrc' (and substitute username & groupname for something appropriate based on the user you created).

```
umask 077
PS1="[\u@\h \W]\\$ "
alias ls='ls -alFG'
```

D. Create a .bash_profile file in root's home directory and enter the following items (as a starting point). After the file has been created, chmod 600 on it so that it's only readable & writable by root. And, just as in the previous step, copy your new .bash_profile to your user's home directory and change the owner on it so that the user owns it (not root).

```
PATH=/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/bin:/usr/local/sbin:$HOM
export PATH
umask 077
PS1="[\u@\h \W]\\$ "
alias ls='ls -alFG'
```

E. Test your settings by going over your 2nd virtual terminal by hitting <Alt>-F2, then logging in as root. Verify that you're using the bash shell, your cursor line looks different (i.e. it has your userid & current working directory), and that you get colorized directory listings. Close out that session and return to your first virtual termial, log out, and then log back in...and then 'su' to root.

144. Redirect root's email to your "normal" email account so that it doesn't get backed up the firewall

A. Use vi to open the /etc/aliases file for editing.

B. Modify line that says "# root: me@my.domain" by removing the "#" comment at the beginning of the line, and then modifying the "me@my.domain" email address so that it points to your "normal" email address instead. You can either point it to your new user account (so that the email stays on the machine & can be accessed without su'ing to root), or redirect it to your 'normal' email account in the office (so that you don't even have to SSH out to the firewall to see how it's doing each day).

C. After saving & exiting, then run the command "newaliases" from the command prompt to update the email alias database.

145. Create & install a warning banner. Use vi to replace your /etc/motd file with the following text (or some other equivalent legal disclaimer). Make sure that you add a line that says 'update_motd="NO"' at the end of your /etc/rc.conf file when you're done...otherwise your changes will be overwritten each time the system reboots.

```
* * * * * * * * * * * * W A R N I N G * * * * * * * * * * * * *
THIS SYSTEM IS RESTRICTED TO AUTHORIZED USERS FOR AUTHORIZED USE
```

```
ONLY. UNAUTHORIZED ACCESS IS STRICTLY PROHIBITED AND MAY BE
PUNISHABLE UNDER THE COMPUTER FRAUD AND ABUSE ACT OF 1986 OR
OTHER APPLICABLE LAWS. IF NOT AUTHORIZED TO ACCESS THIS SYSTEM,
DISCONNECT NOW. BY CONTINUING, YOU CONSENT TO YOUR KEYSTROKES
AND DATA CONTENT BEING MONITORED. ALL PERSONS ARE HEREBY
NOTIFIED THAT THE USE OF THIS SYSTEM CONSTITUTES CONSENT TO
MONITORING AND AUDITING.
* * * * * * * * * * * * W A R N I N G * * * * * * * * * * * * *
```

21. Copy your warning banner over to your /etc/issue file. This will make the warning banner visible at the console before the login prompt...so that people consent to monitoring before they even try to log in:

```
[root@numa /root]# cp /etc/motd /etc/issue
```

22. Configure cvsup and update your source tree & ports collection.

Note #1: After you configure cvsup and update your source and ports collection, you will want to re-run cvsup every once in a while to ensure your sources & ports collection is up-to-date (in case you want to install any new software). Then, you'll want to recompile your kernel & system binaries to ensure you are using the latest versions with security patches applied.

Note #2: We are only updating sections of the ports tree that would be "normal" for a firewall. (i.e. we are not updating the ports collection for games, X-windows, etc.). This will save disk space by not wasting it on ports you won't be installing on a firewall. I'll make the assumption that since you installed packages from the WWW, Mail, Net, and Shells section of the packages collection during the installation (i.e. when you installed lynx, etc.), you'll want the same sections of the ports collection kept up to date. In addition, we'll add to other areas with tools that might be useful on a firewall - the "security" and "sysutils" areas. Add whichever areas you want, but be aware that the more you add...the more hard disk space you'll "eat up." To get a list of which sections of the ports collection are available, do a 'more /usr/share/examples/cvsup/ports-supfile' and browse through the listings of individual ports collection names.

```
[root@numa /root]# cp /usr/src/share/examples/cvsup/stable-supfile
/etc
[root@numa /root]# vi /etc/stable-supfile
```

```
- Run the ":set num" command in vi so that
you can see the line numbers on each line of
the file.
- Change line 66 of the file so that it
points cvsup to a CVS server near you. I
change mine to read '*default
host=cvsup2.FreeBSD.org'. Section A.5.7.
(CVSup Sites) of the FreeBSD Handbook will
tell you where the CVSup servers are.
- On line 71, modify the "tag" variable to
correspond to the specific release of the
O/S that you want to track. The default
value of the tag in the example file is
"RELENG_4". This will download the source
code for the O/S which will has all of the
security updates as well as general bugfixes
and feature enhancements. If, however,
you're in a production environment and can't
afford even the slightest risk of feature
enhancements causing problems with your
production configuration, there's a
different value for this tag that's just for
```

```
you. In this case, set the tag to
"RELENG_4_6". This has ONLY the security
fixes...no feature enhancements...so it's
arguably the more stable version of the
4.6-STABLE branch. 95% of sysadmin's should
change the tag to "RELENG_4_6" to track the
security-related "4.6-STABLE" baseline and
not mess with new enhancements which might
impact the system's stability. It's your
system...it's your call... The official
information about tag was disseminated via
the FreeBSD Security Advisories mailing list
on 11 May 2001 (message subject, "Changes to
FreeBSD security support policy").
- Add these lines at the bottom of the file:
    ports-www tag=.
    ports-mail tag=.
    ports-net tag=.
    ports-shells tag=.
    ports-ftp tag=.
    ports-security tag=.
    ports-sysutils tag=.
- ...and other lines for ports collections
you want...

[root@numa /root]# cvsup /etc/stable-supfile
```

(Source tree is synchronized with CVS server...should take 30-60 minutes...)

38. Configure the SSH daemon and your user's DSA key files.

A. Modify the SSH daemon configuration file, /etc/ssh/sshd_config, so that it reads as follows. The modified lines are in bold red text.

```
# This is ssh server systemwide configuration file. See
sshd(8)
# for more information

Port 22
Protocol 2
ListenAddress 192.168.1.1    *** Put your internal interface's
address here ***
#ListenAddress ::    *** Delete this line ***
HostKey /etc/ssh/ssh_host_key     *** Delete this line ***
HostDsaKey /etc/ssh/ssh_host_dsa_key
ServerKeyBits 768
LoginGraceTime 120
KeyRegenerationInterval 3600
PermitRootLogin no
# ConnectionsPerPeriod has been deprecated completely

# After 3 unauthenticated connections, refuse 50% of the new
ones, and
# refuse any more than 10 total.
MaxStartups 3:50:10
# Don't read ~/.rhosts and ~/.shosts files
IgnoreRhosts yes
# Uncomment if you don't trust ~/.ssh/known_hosts for
RhostsRSAAuthentication
#IgnoreUserKnownHosts yes
StrictModes yes
X11Forwarding no
X11DisplayOffset 10
PrintMotd yes
PrintLastLog yes
KeepAlive yes

# Logging
```

```
SyslogFacility AUTH
LogLevel VERBOSE
#obsoletes QuietMode and FascistLogging

RhostsAuthentication oo
#
# For this to work you will also need host keys in
/etc/ssh_known_hosts
RhostsRSAAuthentication oo
# similar for protocol version 2
HostbasedAuthentication oo
#
RSAAuthentication yes

# To disable tunneled clear text passwords, change to no here!
PasswordAuthentication oo
PermitEmptyPasswords oo

# Uncomment to disable s/key passwords
ChallengeResponseAuthentication oo

# To change Kerberos options
#KerberosAuthentication oo
#KerberosOrLocalPasswd yes
#AFSTokenPassing oo
#KerberosTicketCleanup oo

# Kerberos TGT Passing does only work with the AFS kaserver
#KerberosTgtPassing yes

CheckMail yes
#UseLogin oo

#MaxStartups 10:30:60    *** Delete this line, it's a
duplicate ***
Banner /etc/issue.net
#ReverseMappingCheck yes

Subsystem    sftp   /usr/libexec/sftp-server

AllowUsers oewuser    *** Substitute 'oewuser' with your
userid ***
```

```
[root@numa root]# mount -t msdos /dev/fd0 /mnt/floppy ***
Insert a DOS-formatted floppy before you do this ***
[root@numa root]# cd /mnt/floppy
[root@numa floppy]# cp /home/testuser/.ssh/id_dsa* .  ***
Copies all of your user's ssh key info to the floppy
[root@numa floppy]# ls                                *** List
the contents of the floppy to verify the files are there
[root@numa floppy]# cd ..
[root@numa mnt]# umount /mnt/floppy                   ***
Unmount the floppy
```

E. Now that you've copied your user's private & public keys to another system, remove them from your user's .ssh directory on the firewall. This is only a precaution so that it can't be stolen by a hacker and compromised. If you haven't copied it yet (and plan to do it later), then skip this step until after you've done so.

F. Open up your /etc/hosts.allow file, delete all of the lines, and ensure that it reads as follows. Note that 192.168.1.0 is the address space of your internal network in this example. If you're using a different internal address space (e.g. 10.10.10.0), then make the appropriate modifications.

```
#
# hosts.allow access control file for "tcp wrapped"
applications.
#
ALL : localhost 127.0.0.1 : allow
sshd : 192.168.1.0/255.255.255.0 : allow
ALL : ALL : deny

# If you want to allow a specific computer on the Internet to
SSH into your
# system, replace the 'sshd' line above with one like
this...but subsitute
# the X.X.X.X and subnet mask to suit your needs (e.g. one
computer, entire subnet
# etc.). Also, make sure you allow inbound SSH from that same
host/subnet
# in your /etc/ipf.rules file.
# sshd : 192.168.1.0/255.255.255.0 X.X.X.X/255.255.255.255 :
allow
```

5706 Install and configure Tripwire

A. First, install gmake from the FreeBSD ports collection:

```
[root@numa /root]# cd /usr/ports/devel/gmake
[root@numa gmake]# make && make install
```

B. Download Tripwire-2.3.1-2 from sourceforge.net. If a new version exists, then use it instead. The configuration changes itemized, below, should remain consistent between versions of Tripwire.

```
[root@numa gmake]# cd /root
[root@numa /root]# lynx
http://prdownloads.sourceforge.net/tripwire

          - Use the down-arrow to move through
          the hyperlinks until the file,
          tripwire-2.3.1-2.tar.gz, is
          highlighted, then press [Enter]
          - Use the down-arrow to move through
          the hyperlinks until you've selected a
          mirror site to download from, and then
```

```
arrow over to the "download" hyperlink
for that download site. Hit [Enter]
- When asked if you want to D)ownload
the file, or C)ancel, hit 'd'
- ...file downloads...
- After the file downloads, you'll be
presented with lynx's Download Options
screen. The 'Save to disk' hyperlink is
automatically highlighted in red, so
just hit [Enter].
- Either accept the original filename
by pressing [Enter], or modify the
filename then hit [Enter] to save it.
- After the file is saved, press 'q' to
quit lynx.

[root@numa /root]# tar zxvf tripwire-2.3.1-2.tar.gz
```

C. Modify the Makefile so that it will compile for FreeBSD.

Note: that in several of the following steps, I'll be referring to exact line numbers in the files (some of which are a few hundred lines down). To identify each line of a text file with a line number in vi, use the ":set num" command after you've opened the file. The screen will refresh, and you'll see all of the line numbers down the left side of the screen.

```
[root@numa /root]# cd tripwire-2.3.1-2/src
[root@numa src]# vi Makefile

           - Add a comment at the beginning of
           line 82 (SYSPRE = i686-pc-linux)
           - Remove the '#' comment delimeter at
           the beginning of line 84 (SYSPRE =
           i386-unknown-freebsd)
           - Save and exit.
```

D. Compile Tripwire. On my overclocked Pentium-200MMX, this takes about 45 minutes.

```
[root@numa src]# gmake release
```

E. Configure and install Tripwire

i. Open Tripwire's installation configuration file using vi, and edit it as follows

```
[root@numa src]# cd ../install/
[root@numa install]# vi install.cfg

           - Change line 27 so that it reads
           'TWBIN="/usr/local/sbin"'
           - Change line 33 so that it reads
           'TWMAN="/usr/share/man"'
           - Change line 39 so that it reads
           'TWDOCS="/usr/share/doc/tripwire"'
           - Change line 51 so that it reads
           'TWEDITOR="/usr/bin/vi"'
           - Change line 88 so that it reads
           'TWMAILPROGRAM="/usr/sbin/sendmail
           -oi -t"'
           - Save and exit.
```

ii. Open Tripwire's installation script using vi, and edit it as follows

```
[root@numa install]# vi install.sh
```

```
- Change line 319 so that it
reads
'EULA_PATH="../$TWLICENSEFILE"'
- Change line 491 so that it
reads
'BIN_DIR="../bin/i386-unknown-freebsd_r"'
- Change lines 621-638 so that
they read as follows:
    f1=' ff=$README ; d="/.." ;
dd=$TWDOCS ; rr=0444 '
    f2=' ff=$REL_NOTES ; d="/.."
; dd=$TWDOCS ; rr=0444 '
    f3=' ff=$TWLICENSEFILE ;
d="/.." ; dd=$TWDOCS ; rr=0444 '
    f4=' ff=tripwire ;
d="/../bin/i386-unknown-freebsd_r"
; dd=$TWBIN ; rr=0550 '
    f5=' ff=twadmin ;
d="/../bin/i386-unknown-freebsd_r"
; dd=$TWBIN ; rr=0550 '
    f6=' ff=twprint ;
d="/../bin/i386-unknown-freebsd_r"
; dd=$TWBIN ; rr=0550 '
    f7=' ff=siggen ;
d="/../bin/i386-unknown-freebsd_r"
; dd=$TWBIN ; rr=0550 '
    f8=' ff=TRADEMARK ; d="/.." ;
dd=$TWDOCS ; rr=0444 '
    f9=' ff=policyguide.txt ;
d="/../policy" ; dd=$TWDOCS ;
rr=0444 '
    f10=' ff=twpol.txt ;
d="/../policy" ; dd=$TWPOLICY ;
rr=0640 '
    f11=' ff=twpolicy.4 ;
d="/../man/man4" ; dd=$TWMAN/man4
; rr=0444 '
    f12=' ff=twconfig.4 ;
d="/../man/man4" ; dd=$TWMAN/man4
; rr=0444 '
    f13=' ff=twfiles.5 ;
d="/../man/man5" ; dd=$TWMAN/man5
; rr=0444 '
    f14=' ff=siggen.8 ;
d="/../man/man8" ; dd=$TWMAN/man8
; rr=0444 '
    f15=' ff=tripwire.8 ;
d="/../man/man8" ; dd=$TWMAN/man8
; rr=0444 '
    f16=' ff=twadmin.8 ;
d="/../man/man8" ; dd=$TWMAN/man8
; rr=0444 '
    f17=' ff=twintro.8 ;
d="/../man/man8" ; dd=$TWMAN/man8
; rr=0444 '
    f18=' ff=twprint.8 ;
d="/../man/man8" ; dd=$TWMAN/man8
; rr=0444 '
- Save and exit.
```

iii. Install Tripwire

```
[root@numa install]# ./install.sh

- Answer 'y' to continue with the
installation
- Press [Enter] to view the
license agreement...when
complete, type 'accept' and
[Enter]
- The install script will verify
that sendmail and vi are
```

```
                    installed, then verify that the
                    tripwire binaries are available,
                    and then echo back all of the
                    configuration parameters for the
                    installation script (e.g. TWBIN,
                    TWMAN, etc.). If everything looks
                    good, answer 'y' to continue with
                    the installation.
                    - The install script copies all
                    of the files, the asks you to
                    enter a new site keyfile
                    passphrase. Enter it, and then
                    enter it again when asked to
                    verify it.
                    - The install script then asks
                    you to enter a new local keyfile
                    passphrase. Enter it, and then
                    enter it again when asked to
                    verify it.
                    - The install script will then
                    create a signed configuration
                    file, but will need you to enter
                    the site passphrase you just set,
                    above. Enter it.
                    - The install script will then
                    create a signed policy file, but
                    will need you to enter the site
                    passphrase you just set, above.
                    Enter it.
                    - ...installation is complete.
```

F. Install a new Tripwire text policy file. Replace the contents of the file /etc/tripwire/twpol.txt with the following. It is a functional Tripwire policy configuration file for FreeBSD-4.3, but feel free to edit it based upon your own special installation & configuration. Note that you'll have to modify the two items in bold red text to match your configuration (i.e. your system's hostname and your non-priveleged username):

```
@@section GLOBAL
TWROOT="/usr/local";
TWBIN="/usr/local/sbin";
TWPOL="/etc/tripwire";
TWDB="/var/lib/tripwire";
TWSKEY="/etc/tripwire";
TWLKEY="/etc/tripwire";
TWREPORT="/var/lib/tripwire/report";
HOSTNAME=hostname.domain;

@@section FS
SEC_CRIT = $(IgnoreNone)-SHa; # Critical files - we can't
afford to miss any changes.
SEC_SUID = $(IgnoreNone)-SHa; # Binaries with the SUID or SGID
flags set.
SEC_TCB = $(ReadOnly); # Members of the Trusted Computing
Base.
SEC_BIN = $(ReadOnly); # Binaries that shouldn't change
SEC_CONFIG = $(Dynamic); # Config files that are changed
infrequently but accessed often.
SEC_LOG = $(Growing); # Files that grow, but that should never
change ownership.
SEC_INVARIANT = +pug; # Directories that should never change
permission or ownership.
SIG_LOW = 33; # Non-critical files that are of minimal
security impact
SIG_MED = 66; # Non-critical files that are of significant
security impact
SIG_HI = 100; # Critical files that are significant points of
vulnerability
```

```
# Tripwire Binaries
(rulename = "Tripwire Binaries", severity = $(SIG_HI))
{
  $(TWBIN)/siggen -> $(SEC_TCB);
  $(TWBIN)/tripwire -> $(SEC_TCB);
  $(TWBIN)/twadmin -> $(SEC_TCB);
  $(TWBIN)/twprint -> $(SEC_TCB);
}

# Tripwire Data Files - Configuration Files, Policy Files,
Keys, Reports, Databases
(rulename = "Tripwire Data Files", severity = $(SIG_HI))
{
  # NOTE: Removing the inode attribute because when Tripwire
creates a backup
  # it does so by renaming the old file and creating a new one
(which will
  # have a new inode number). Leaving inode turned on for
keys, which shouldn't
  # ever change.

  # NOTE: this rule will trigger on the first integrity check
after database
  # initialization, and each integrity check afterward until a
database update
  # is run, since the database file will not exist before that
point.
  $(TWDB) -> $(SEC_CONFIG) -i;
  $(TWPOL)/tw.pol -> $(SEC_BIN) -i;
  $(TWPOL)/tw.cfg -> $(SEC_BIN) -i;
  $(TWLKEY)/$(HOSTNAME)-local.key -> $(SEC_BIN);
  $(TWSKEY)/site.key -> $(SEC_BIN);

  #don't scan the individual reports
  $(TWREPORT) -> $(SEC_CONFIG) (recurse=0);
}

# These files are critical to a correct system boot.
(rulename = "Critical system boot files", severity = 100)
{
  /boot -> $(SEC_CRIT);
  /kernel -> $(SEC_CRIT);
}

# These files change the behavior of the root account and also
the authorized_keys2
# file for the user we created earlier
(rulename = "Root config files", severity = 100)
{
  /root -> $(SEC_CRIT);
  /root/.bash_history -> $(SEC_LOG);
  /root/.bash_profile -> $(SEC_CRIT);
  /root/.bashrc -> $(SEC_CRIT);
  /home/username/.ssh/authorized_keys2 -> $(SEC_CRIT);
}

# Commonly accessed directories that should remain static with
regards to owner and group
(rulename = "Invariant Directories", severity = $(SIG_MED))
{
  / -> $(SEC_INVARIANT) (recurse = 0);
  /etc -> $(SEC_INVARIANT) (recurse = 0);
  /usr/local/etc -> $(SEC_INVARIANT) (recurse = 0);
}

(rulename = "Shell Binaries", severity = $(SIG_HI))
{
  /usr/local/bin/bash -> $(SEC_BIN);
  /bin/csh -> $(SEC_BIN);
  /bin/sh -> $(SEC_BIN);
  /bin/tcsh -> $(SEC_BIN);
```

```
}

# Rest of critical system binaries
(rulename = "OS executables and libraries", severity =
$(SIG_HI))
{
  /bin -> $(SEC_BIN) (recurse = 1);
  /usr/bin -> $(SEC_BIN) (recurse = 1);
  /usr/lib -> $(SEC_BIN) (recurse = 1);
  /sbin -> $(SEC_BIN) (recurse = 1);
  /usr/sbin -> $(SEC_BIN) (recurse = 1);
}

# Local files
(rulename = "User executables and libraries", severity =
$(SIG_MED))
{
  /usr/local/bin -> $(SEC_BIN) (recurse = 1);
  /usr/local/sbin -> $(SEC_BIN) (recurse = 1);
}

# Temporary directories
(rulename = "Temporary directories", recurse = false, severity
= $(SIG_LOW))
{
  /usr/tmp -> $(SEC_INVARIANT);
  /var/tmp -> $(SEC_INVARIANT);
  /tmp -> $(SEC_INVARIANT);
}

# Include
(rulename = "OS Development Files", severity = $(SIG_MED))
{
  /usr/include -> $(SEC_BIN);
  /usr/local/include -> $(SEC_BIN);
}

# Shared
(rulename = "OS Shared Files", severity = $(SIG_MED))
{
  /usr/share -> $(SEC_BIN);
  !/usr/share/man;
  /usr/local/share -> $(SEC_BIN);
}

# setuid/setgid root programs
(rulename = "setuid/setgid", severity = $(SIG_HI))
{
  /bin/df -> $(SEC_SUID);
  /bin/rcp -> $(SEC_SUID);
  /sbin/ccdconfig -> $(SEC_SUID);
  /sbin/dmesg -> $(SEC_SUID);
  /sbin/dump -> $(SEC_SUID);
  /sbin/ping -> $(SEC_SUID);
  /sbin/ping6 -> $(SEC_SUID);
  /sbin/rdump -> $(SEC_SUID);
  /sbin/restore -> $(SEC_SUID);
  /sbin/route -> $(SEC_SUID);
  /sbin/rrestore -> $(SEC_SUID);
  /sbin/shutdown -> $(SEC_SUID);
  /usr/bin/at -> $(SEC_SUID);
  /usr/bin/atq -> $(SEC_SUID);
  /usr/bin/atrm -> $(SEC_SUID);
  /usr/bin/batch -> $(SEC_SUID);
  /usr/bin/chfn -> $(SEC_SUID);
  /usr/bin/chpass -> $(SEC_SUID);
  /usr/bin/chsh -> $(SEC_SUID);
  /usr/bin/crontab -> $(SEC_SUID);
  /usr/bin/cu -> $(SEC_SUID);
  /usr/bin/fstat -> $(SEC_SUID);
  /usr/bin/ipcs -> $(SEC_SUID);
  /usr/bin/keyinfo -> $(SEC_SUID);
```

```
  /usr/bin/keyinit -> $(SEC_SUID);
  /usr/bin/lock -> $(SEC_SUID);
  /usr/bin/login -> $(SEC_SUID);
  /usr/bin/lpq -> $(SEC_SUID);
  /usr/bin/lpr -> $(SEC_SUID);
  /usr/bin/lprm -> $(SEC_SUID);
  /usr/bin/man -> $(SEC_SUID);
  /usr/bin/netstat -> $(SEC_SUID);
  /usr/bin/nfsstat -> $(SEC_SUID);
  /usr/bin/passwd -> $(SEC_SUID);
  /usr/bin/quota -> $(SEC_SUID);
  /usr/bin/rlogin -> $(SEC_SUID);
  /usr/bin/rsh -> $(SEC_SUID);
  /usr/bin/su -> $(SEC_SUID);
  /usr/bin/systat -> $(SEC_SUID);
  /usr/bin/top -> $(SEC_SUID);
  /usr/bin/uucp -> $(SEC_SUID);
  /usr/bin/uuname -> $(SEC_SUID);
  /usr/bin/uustat -> $(SEC_SUID);
  /usr/bin/uux -> $(SEC_SUID);
  /usr/bin/vmstat -> $(SEC_SUID);
  /usr/bin/wall -> $(SEC_SUID);
  /usr/bin/write -> $(SEC_SUID);
  /usr/bin/ypchfn -> $(SEC_SUID);
  /usr/bin/ypchpass -> $(SEC_SUID);
  /usr/bin/ypchsh -> $(SEC_SUID);
  /usr/bin/yppasswd -> $(SEC_SUID);
  /usr/libexec/sendmail/sendmail -> $(SEC_SUID);
  /usr/libexec/uucp/uucico -> $(SEC_SUID);
  /usr/libexec/uucp/uuxqt -> $(SEC_SUID);
  /usr/local/bin/mutt_dotlock -> $(SEC_SUID);
  /usr/sbin/ifmcstat -> $(SEC_SUID);
  /usr/sbin/iostat -> $(SEC_SUID);
  /usr/sbin/lpc -> $(SEC_SUID);
  /usr/sbin/mrinfo -> $(SEC_SUID);
  /usr/sbin/mtrace -> $(SEC_SUID);
  /usr/sbin/ppp -> $(SEC_SUID);
  /usr/sbin/pppd -> $(SEC_SUID);
  /usr/sbin/pstat -> $(SEC_SUID);
  /usr/sbin/sliplogin -> $(SEC_SUID);
  /usr/sbin/swapinfo -> $(SEC_SUID);
  /usr/sbin/timedc -> $(SEC_SUID);
  /usr/sbin/traceroute -> $(SEC_SUID);
  /usr/sbin/traceroute6 -> $(SEC_SUID);
  /usr/sbin/trpt -> $(SEC_SUID);
}

(rulename = "Configuration Files", severity = $(SIG_MED))
{
  /etc/hosts -> $(SEC_CONFIG);
  /etc/inetd.conf -> $(SEC_CONFIG);
  /etc/resolv.conf -> $(SEC_CONFIG);
  /etc/syslog.conf -> $(SEC_CONFIG);
  /etc/newsyslog.conf -> $(SEC_CONFIG);
}

(rulename = "Security Control", severity = $(SIG_HI))
{
  /etc/group -> $(SEC_CRIT);
}

(rulename = "Login Scripts", severity = $(SIG_HI))
{
  /etc/csh.login -> $(SEC_CONFIG);
  /etc/csh.logout -> $(SEC_CONFIG);
  /etc/csh.cshrc -> $(SEC_CONFIG);
  /etc/profile -> $(SEC_CONFIG);
}

# These files change every time the system boots
(rulename = "System boot changes", severity = $(SIG_HI))
{
```

```
  /dev/log -> $(Dynamic);
  /dev/cuaa0 -> $(Dynamic);
  /dev/console -> $(Dynamic);
  /dev/ttyv0 -> $(Dynamic);
  /dev/ttyv1 -> $(Dynamic);
  /dev/ttyv2 -> $(Dynamic);
  /dev/ttyv3 -> $(Dynamic);
  /dev/ttyv4 -> $(Dynamic);
  /dev/ttyv5 -> $(Dynamic);
  /dev/ttyv6 -> $(Dynamic);
  /dev/ttyp0 -> $(Dynamic);
  /dev/ttyp1 -> $(Dynamic);
  /dev/ttyp2 -> $(Dynamic);
  /dev/ttyp3 -> $(Dynamic);
  /dev/ttyp4 -> $(Dynamic);
  /dev/ttyp5 -> $(Dynamic);
  /dev/ttyp6 -> $(Dynamic);
  /dev/urandom -> $(Dynamic);
}

# Critical configuration files
(rulename = "Critical configuration files", severity =
$(SIG_HI))
{
  /etc/crontab -> $(ReadOnly);
  /etc/periodic/daily -> $(ReadOnly);
  /etc/periodic/weekly -> $(ReadOnly);
  /etc/periodic/monthly -> $(ReadOnly);
  /etc/periodic/security -> $(ReadOnly);
  /etc/defaults -> $(ReadOnly);
  /etc/fstab -> $(ReadOnly);
  /etc/hosts.allow -> $(ReadOnly);
  /etc/ttys -> $(ReadOnly);
  /etc/gettytab -> $(ReadOnly);
  /etc/protocols -> $(ReadOnly);
  /etc/services -> $(ReadOnly);
  /etc/rc -> $(ReadOnly);
  /etc/rc.conf -> $(ReadOnly);
  /etc/rc.atm -> $(ReadOnly);
  /etc/rc.diskless1 -> $(ReadOnly);
  /etc/rc.diskless2 -> $(ReadOnly);
  /etc/rc.firewall -> $(ReadOnly);
  /etc/rc.firewall6 -> $(ReadOnly);
  /etc/rc.i386 -> $(ReadOnly);
  /etc/rc.isdn -> $(ReadOnly);
  /etc/rc.network -> $(ReadOnly);
  /etc/rc.network6 -> $(ReadOnly);
  /etc/rc.pccard -> $(ReadOnly);
  /etc/rc.resume -> $(ReadOnly);
  /etc/rc.serial -> $(ReadOnly);
  /etc/rc.shutdown -> $(ReadOnly);
  /etc/rc.suspend -> $(ReadOnly);
  /etc/rc.syscons -> $(ReadOnly);
  /etc/rc.sysctl -> $(ReadOnly);
  /etc/motd -> $(ReadOnly);
  /etc/passwd -> $(ReadOnly);
  /etc/master.passwd -> $(ReadOnly);
  /etc/pwd.db -> $(ReadOnly);
  /etc/spwd.db -> $(ReadOnly);
  /etc/rpc -> $(ReadOnly);
  /etc/shells -> $(ReadOnly);
  /etc/ipf.rules -> $(ReadOnly);
  /etc/ipnat.rules -> $(ReadOnly);
  /etc/ssh/sshd_config -> $(ReadOnly);
}

# Critical devices
(rulename = "Critical devices", severity = $(SIG_HI), recurse
= false)
{
  /dev/kmem -> $(Device);
  /dev/mem -> $(Device);
```

```
  /dev/null -> $(Device);
  /dev/zero -> $(Device);
}
```

G. Re-generate the Tripwire policy file and database - Now that you have a good Tripwire text policy file, we need to actually create the policy file from it and the Tripwire database itself. To do that, just type the following commands:

```
[root@numa /root]# twadmin --create-polfile --cfgfile
/etc/tripwire/tw.cfg /etc/tripwire/twpol.txt
[root@numa /root]# tripwire --init --cfgfile
/etc/tripwire/tw.cfg

     *** Note: You will receive an error that says that two
files do
         not exist yet. These two files are /etc/ipf.rules and
/etc/ipnat.rules.
         That's OK because we haven't created them yet. We
won't until another
         5-10 steps from now. After you have completed this
HOWTO, simply
         re-initialize the tripwire database & everything will
be OK.
```

H. Create a cron job to check the integrity of your system every day at 4AM:

```
[root@numa /root]# cd /etc
[root@numa /etc]# vi crontab

      - Add the following line to the file:
         0   4   *   *   *
         root   /usr/local/sbin/tripwire --check
      --cfgfile /etc/tripwire/tw.cfg
```

I. In the future, when you want to interactively check your system for changes, and then incorporate those changes into the Tripwire database, run the following command. After you run it, you'll be presented with a report of the policy violations. Edit it with vi, leaving an 'x' next to each policy violation that you want the database updated with. Then edit & save.

```
[root@numa /root]# tripwire --check --interactive --cfgfile
/etc/tripwire/tw.cfg
```

9250. Edit your /etc/rc.conf file so that it's ready for the screen resolution changes and the IPFILTER modifications we'll make in a few minutes.

A. Towards the end of the file, there's a line that reads 'sendmail_enable="YES"'. Change it so that it reads 'sendmail_enable="NONE"'.

B. Add the following lines at the bottom of the file to support 132x43 screen resolution (after we compile VESA support into the kernel, below).

```
font8x8="/usr/share/syscons/fonts/iso02-8x8.fnt"
allscreens_flags="132x43"
```

C. Add the following line at the bottom of the file so that syslog won't log to remote machines, nor will it accept logs from remote machines. This still allows syslog to function, but stops it from being a 'listening' service.

```
syslogd_flags="-ss"
```

D. Add the following line at the bottom of the file so that SSHD only listens for IPv4 addresses

```
sshd_flags="-4"
```

E. Add the following lines at the bottom of the file so that IPFILTER, IPNAT, and IPMON will work correctly after we compile support for it into the kernel and create the appropriate files, below. The forced use of no options for ipfilter overrides the default "-E" flag in /etc/defaults/rc.conf. By overriding this option, you won't get errors when ipfilter starts up that complain that ipfilter is already running. The options for ipmon perform the following - D causes it to run as a daemon, s tells it to log to syslog rather than a file, v tells it to log the tcp window, ack and sequence fields, and n tells it to map the IP addresses and port numbers back to hostnames and service names.

```
ipfilter_enable="YES"
ipfilter_flags=""
ipmon_enable="YES"
ipmon_flags="-Dsvn"
ipnat_enable="YES"
```

F. Modify the following line so that your 2nd ISA network card is a valid network interface. Sometimes, this line may not be present in the /etc/rc.conf file. If it's not, then add it. Just a reminder that the device names I'm using here (ed0 & ed1) are for NE2000-compatible ISA cards. If you're using PCI cards, the device names will be different.

```
network_interfaces="ed0 ed1 lo0"
```

G. Add the following line so that your new 2nd ISA network card is configured correctly after you recompile the kernel and reboot. Again, we're assuming that you're using 192.168.1.0 as the internal network (per the diagram at the beginning of this document):

```
ifconfig_ed1="inet 192.168.1.1 netmask 255.255.255.0"
```

H. Ensure the following two lines are present. If they're not, then add them. Again, per the diagram at the beginning of this document, ed0 is your DHCP interface connected to your ISP. And lo0 (as always) is your loopback address).

```
ifconfig_lo0="inet 127.0.0.1"
ifconfig_ed0="DHCP"
```

I. Reduce your vulnerability to Denial of Service (DoS) or hijack attacks by dropping ICMP redirect packets. Add the following line to the bottom of the file:

```
icmp_drop_redirects="YES"
```

9403. Create a separate logfile for our firewall logs and another one for our SSHD authentication log entries. Then edit the newsyslog configuration file so that your new logfiles are rotated properly.

A. Create a new file for the firewall and authentication logs with the following commands:

```
[root@numa /root]# touch /var/log/firewall_logs
[root@numa /root]# touch /var/log/authlog
[root@numa /root]# chmod 600 /var/log/firewall_logs
[root@numa /root]# chmod 600 /var/log/authlog
```

B. Modify your syslog configuration file (/etc/syslog.conf) so that the IPFILTER logged events (logged with IPMON) are sent to your new separate firewall log file. Insert the following line at the top of the file:

```
local0.*     /var/log/firewall_logs
```

Then, add the entry "local0.none" to the line for /var/log/messages. Add it to the middle of the line, separated from the other entries with a semi-colon. This will ensure that the firewall log entries don't end up in /var/log/messages...they'll only go to your firewall log, configured above.

C. Insert the following line towards the top of the file so that the SSHD

logged events are sent to a log file called "authlog":

```
auth.*      /var/log/authlog
```

D. Modify your newsyslog configuration file (/etc/newsyslog.conf) so that your new firewall log files get rotated just like the primary syslog file (/var/log/messages). Add the following new line to the bottom of the file:

```
/var/log/firewall_logs    600   14   100   *   Z
/var/log/authlog          600   14   100   *   Z
```

9507. Create your IPFILTER and IPNAT rulesets

A. Using vi, create a new IPFILTER firewall ruleset, /etc/ipf.rules, & add the following lines to it. Note: The assumption is that ed0 is the "outside" interface (i.e. connected to your ISP), and ed1 is the "inside" interface (i.e. connected to your internal network). Also note that we're not performing egress filtering here. We're blocking all inbound packets from the internet and allowing all internal network packets out (and keeping state on them so that they're allowed back in). After your box is configured to your liking, I heavily recommend implementing egress filtering.

For those new to egress filtering...all it means is that you only allow out of your network traffic that you explicitly want to let out. For example, you'd change the line that allows unrestricted outbound tcp traffic (the first rule in the ruleset) into 5 or more different rules. One which allows outbound traffic as long as it's going to port 80 (http). The second allows outbound traffic as long as it's going to port 25 (smtp), etc. Add as many rules as you need to define the outbound traffic that you're allowing. Then, add a rule before all of these that blocks all outbound traffic to broadcast addresses (i.e. anything that ends with a 255...like x.x.x.255). And you'd add another rule that blocks all outbound traffic if the source address isn't on the 192.168.1.0 network or the IP address of your ed0 interface (the one that's connected to your ISP). So, you'd be blocking all packets that aren't coming from your network or your own system. In other words, you know that your users will only need to go out to web sites, send mail, etc. And they'll never need to send broadcast packets out to the Internet, etc. And they better not be spoofing their source IP address. This is only a sample of what egress filtering is all about. Good (read as 'restrictive') egress filtering can be quite complex, but is in the best interest of the Internet because it doesn't allow your box (in the off-chance that it does get hacked) to be used maliciously for things like "smurf" attacks and other broadcast amplification attacks (where your system sends out broadcast packets to a target network to get as many systems to respond as possible...which eats up their bandwidth). In addition, it lets you know if you have any systems on the inside of your network that are trying to access the internet over unauthorized protocols & services (read as "misbehaving users...").

And as a final note, since we're using IPFILTER's stateful packet inspection abilities, we don't need to reject traffic spoofing non-routable or reserved addresses...they'll be blocked automatically since they don't match a corresponding packet in the state table. If you do allow certain services into your firewall (say, SSH access from the Internet so that you can manage the firewall remotely), then you'll have to add these filters in. To do so, block all incoming traffic on your ed0 interface that claims to have a source IP address of 192.168.0.0/16, 10.0.0.0/8, or any of the other reserved addresses, etc.

Use this IPFILTER ruleset as a starting point. After you have everything running, add in whatever you want (egress filtering, protection from non-routable addresses, IP spoofing protection, etc.) to complete the job. This is only a starting point.

Note: Remember to modify the bold red text (below) so that it matches the IP address of your ISP's DHCP server

```
##############################################################
# Outside Interface
```

```
##########################################################################

#-------------------------------------------------------------------
# Allow out all TCP, UDP, and ICMP traffic & keep state on it
# so that it's allowed back in.
#
# If you wanted to do egress filtering...here's where you'd do
it.
# You'd change the lines below so that rather than allowing
out any
# arbitrary TCP connection, it would only allow out mail,
pop3, and http
# connections (for example). So, the first line, below, would
be
# replaced with:
#    pass out quick on ed0 proto tcp from any to any port = 25
keep state
#    pass out quick on ed0 proto tcp from any to any port =
110 keep state
#    pass out quick on ed0 proto tcp from any to any port = 80
keep state
# ...and then do the same for the remaining lines so that you
allow
# only specified protocols/ports 'out' of your network
#-------------------------------------------------------------------
pass out quick on ed0 proto tcp from any to any keep state
pass out quick on ed0 proto udp from any to any keep state
pass out quick on ed0 proto icmp from any to any keep state
block out quick on ed0 all

#-------------------------------------------------------------------
# Block all inbound traffic from non-routable or reserved
address spaces
#-------------------------------------------------------------------
block in log quick on ed0 from 192.168.0.0/16 to any  #RFC
1918 private IP
block in log quick on ed0 from 172.16.0.0/12 to any   #RFC
1918 private IP
block in log quick on ed0 from 10.0.0.0/8 to any      #RFC
1918 private IP
block in log quick on ed0 from 127.0.0.0/8 to
any     #loopback
block in log quick on ed0 from 0.0.0.0/8 to
any       #loopback
block in log quick on ed0 from 169.254.0.0/16 to any  #DHCP
auto-config
block in log quick on ed0 from 192.0.2.0/24 to
any    #reserved for doc's
block in log quick on ed0 from 204.152.64.0/23 to any #Sun
cluster interconnect
block in quick on ed0 from 224.0.0.0/3 to any         #Class D
& E multicast

#-------------------------------------------------------------------
# Allow bootp traffic in from your ISP's DHCP server only.
#-------------------------------------------------------------------
pass in quick on ed0 proto udp from X.X.X.X/32 to any port =
68 keep state

#-------------------------------------------------------------------
# If you wanted to set up a web server or mail server on your
box
# (which is outside the scope of this howto), or allow another
system
# on the Internet to externally SSH into your firewall, you'd
want to
# uncomment the following lines and modify as appropriate. If
you
# have other services running that you need to allow external
access
# to, just add more lines using these as examples.
#
```

```
# If the services are on a box on your internal network
(rather than
# the firewall itself), you'll have to add both the filter
listed below,
# plus a redirect rule in your /etc/ipnat.rules file.
#-------------------------------------------------------------
# pass in quick on ed0 proto tcp from any to any port = 80
flags S keep state keep frags
# pass in quick on ed0 proto tcp from any to any port = 25
flags S keep state keep frags
# pass in quick on ed0 proto tcp from X.X.X.X/32 to any port =
22 flags S keep state keep frags

#-------------------------------------------------------------
# Block and log all remaining traffic coming into the firewall
# - Block TCP with a RST (to make it appear as if the service
# isn't listening)
# - Block UDP with an ICMP Port Unreachable (to make it appear

# as if the service isn't listening)
# - Block all remaining traffic the good 'ol fashioned way
#-------------------------------------------------------------
block return-rst in log quick on ed0 proto tcp from any to any
block return-icmp-as-dest(port-unr) in log quick on ed0 proto
udp from any to any
block in log quick on ed0 all

#############################################################
# Inside Interface
#############################################################

#-------------------------------------------------------------
# Allow out all TCP, UDP, and ICMP traffic & keep state
#-------------------------------------------------------------
pass out quick on ed1 proto tcp from any to any keep state
pass out quick on ed1 proto udp from any to any keep state
pass out quick on ed1 proto icmp from any to any keep state
block out quick on ed1 all

#-------------------------------------------------------------
# Allow in all TCP, UDP, and ICMP traffic & keep state
#-------------------------------------------------------------
pass in quick on ed1 proto tcp from any to any keep state
pass in quick on ed1 proto udp from any to any keep state
pass in quick on ed1 proto icmp from any to any keep state
block in quick on ed1 all

#############################################################
# Loopback Interface
#############################################################

#-------------------------------------------------------------
# Allow everything to/from your loopback interface so you
# can ping yourself (e.g. ping localhost)
#-------------------------------------------------------------
pass in quick on lo0 all
pass out quick on lo0 all
```

D. Using vi, create a new IPNAT translation ruleset, /etc/ipnat.rules, & add the following lines to it.

```
#-------------------------------------------------------------------
# Do 'normal' IP address translation. This line will take all
packets
# going out on your external NIC (ed0) that have a source
address coming
# from your internal network (192.168.1.0), and translate it
to whatever
# IP address your external NIC happens to have at that time
#-------------------------------------------------------------------
map ed0 192.168.1.0/24 -> 0/32
```

```
#--------------------------------------------------------------------
# If you have a system on your internal network that needs to
be
# 'reachable' by external systems on the internet, you'll need
a rule
# similar to the one below. This one takes all inbound http
traffic
# (TCP port 80) that hits the firewall's external interface
(ed0) and
# redirects it to port 80 on the 192.168.1.50 system on the
internal network.
# Simply uncomment the rule, change the IP address and port
number so that
# it does what you need. Remember that you have to enable the
corresponding
# inbound filter in your /etc/ipf.rules file, too.
#--------------------------------------------------------------------
# rdr ed0 0.0.0.0/0 port 80 -> 192.168.1.50 port 80 tcp
```

9654 Add "blackhole" support and restrict crontab access/usage

A. Add "blackhole" support to make your system harder to port-scan. Edit your /etc/sysctl.conf file and add the following 2 lines to the end of the file. This will make it much harder for people to port-scan your system by not sending back telltale RST packets to closed TCP ports on your system (for example). By setting the TCP blackhole parameter to "2" it will drop unsolicited packets of any type (SYN, FIN, etc.) to closed ports. Setting the UDP blackhole parameter to "1" will perform the same function for UDP (since UDP isn't connection oriented).

```
net.inet.tcp.blackhole=2
net.inet.udp.blackhole=1
```

B. Create the file /var/cron/allow and add the following lines to it. Be sure to substitute 'newuser' for whatever your non-privileged user account is. This will only allow the root user and your non-privileged user account to schedule jobs with cron.

```
root
newuser
```

C. Edit /etc/crontab and remove the 'at' job that runs every 5 minutes. The 'at' command is run commands at a later time (i.e. type in the command now, and it'll run 20 minutes later, etc.) If you don't use 'at' command (and chances are, you don't), then you don't need it running every 5 minutes. Hackers might want to use it to schedule a job to help give them access to your system or download rootkits, etc. So, edit your /etc/crontab file, and place a '#' at the beginning of the following line (i.e. comment it out).

```
*/5 * * * * root /usr/libexec/atrun
```

D. Chmod your /etc/crontab file so that it is only readable by root. There's no need for other users to know what jobs are scheduled to run & when. It's a firewall, not a common-use workstation.

```
[root@numa /etc]# chmod 600 /etc/crontab
```

9655 Re-compile your kernel & system binaries (i.e. add IPFILTER support, upgrade to -STABLE, add VESA support, and add support for a 2nd ISA Ethernet card)

A. Change directory into /usr/src/sys/i386/conf

```
cd /usr/src/sys/i386/conf
```

B. Copy the file GENERIC to a new file - typically named after your hostname (I'll assume that your hostname is "FIREWALL")

```
cp GENERIC FIREWALL
```

C. Using vi, edit your new file, FIREWALL, and make the following changes:

i. In line 2 of the file (part of the main comment block) change the word, GENERIC, to your hostname, FIREWALL.

ii. On line 18 of the file (still part of the main comment block), change the word, GENERIC, to your hostname, FIREWALL

iii. On lines 21-24, comment out the "cpu" lines so that only the one for your specific chip is left. For a Pentium MMX, I commented out all of them except line 23 - cpu "I586_CPU"

iv. On line 25, change the value of the ident parameter so that it's your hostname, FIREWALL

v. Starting at about line 32 (right after the line with "options INET"), add the following 3 new lines to add IPFILTER support into your firewall and have it automatically block all packets by default:

```
options IPFILTER
options IPFILTER_LOG
options IPFILTER_DEFAULT_BLOCK
```

vi. At about line 135 (immediately following the line "device vga0 at isa?"), add the following line to add support for VESA video modes (for 132x43 resolution):

```
options VESA
```

vii. At about line 200, (in the networking portion...right after the title "ISA Ethernet NICs" and after the line with "device ed0 at isa? port 0x280 irq 10 iomem 0xd8000"), add the following new line to add support for a 2nd generic NE2000-compatible ISA Ethernet card. Pre-configure the IRQ and port to whatever your 2nd ISA Ethernet card is configured to (to save you a little time after you reboot):

```
device ed1 at isa? port 0x280 irq 10 iomem 0xd8000
```

D. After saving your kernel configuration file, FIREWALL, type the following commands in this order to compile your kernel (assuming your new kernal name is "FIREWALL"), as well as all system binaries, and install them. With my old 200MHz Pentium MMX (overclocked to 225MHz) with 96MB of RAM, the "make buildworld" command took 3 hours (179 minutes) since it rebuilds all of the system binaries; the "make kernel" command took 30 minutes; and the "make installworld" command took 9 minutes.

```
[root@numa /]# cd /usr/src
[root@numa src]# echo "KERNCONF=FIREWALL" >> /etc/make.conf
[root@numa src]# make world
[root@numa src]# make kernel
```

9668 Next, edit your /etc/rc.conf file with vi and change the kernel security level to "2" (which is the 'Extreme' setting we mentioned all of the way at the beginning of this HOWTO) by doing the following:

```
- Modify the line that reads 'kern_securelevel_enable="NO"' and
change the value to "YES"
- Add a line beneath it that reads 'kern_securelevel="2"'
```

9669 Lastly, modify the /etc/fstab file with vi so that we can change how each partition is mounted...to ensure that hackers can do at little as possible if they (by chance alone) hack the box. Essentially, we're restricting some of the partitions so that they are 'nosuid', 'noexec', and 'ro'. The original /etc/fstab should look something like this. Yours might look a little different...the first column (device names) might be a little different, but that's OK. The stuff we'll be modifying is in the 4th column.

```
# Device     Mountpoint   FStype   Options   Dump   Pass#
/dev/ad0s1b  none         swap     sw        0      0
/dev/ad0s1a  /            ufs      rw        1      1
```

```
/dev/ad0s1e  /tmp         ufs      rw        2      2
/dev/ad0s1g  /usr         ufs      rw        2      2
/dev/ad0s1d  /usr/home    ufs      rw        2      2
/dev/ad0s1h  /usr/local   ufs      rw        2      2
/dev/ad0s1f  /var         ufs      rw        2      2
proc         /proc        procfs   rw        0      0
```

First, copy the original /etc/fstab file to /etc/fstab.original

Then, make another copy of the /etc/fstab file and call it /etc/fstab.restrictive

Then, modify the /etc/fstab.restrictive file so that it reads as follows:

```
# Device     Mountpoint   FStype   Options                 Dump
  Pass#
/dev/ad0s1b  none         swap     sw                      0      0
/dev/ad0s1a  /            ufs      rw,nosuid               1      1
/dev/ad0s1e  /tmp         ufs      rw,noexec,nosuid,nodev  2      2
/dev/ad0s1g  /usr         ufs      ro                      2      2
/dev/ad0s1d  /usr/home    ufs      rw,noexec,nosuid        2      2
/dev/ad0s1h  /usr/local   ufs      ro,nosuid               2      2
/dev/ad0s1f  /var         ufs      rw,noexec,nosuid        2      2
proc         /proc        procfs   rw                      0      0
```

Next, copy your new /etc/fstab.restrictive file and over-write the original /etc/fstab...so that your "real" fstab file has the restrictive settings, and you have the two other config files available (the original and restrictive one).

```
[root@numa etc]# cp /etc/fstab.restrictive /etc/fstab
```

Note that this will make adding new software, etc. much more difficult since /usr and /usr/local are mounted read-only. This means that programs which try to install their user-land programs in /usr/local/bin will fail during their install programs. And cvsup...which will try to update the kernel's source code in /usr/src and the ports in /usr/ports...well, they're now read-only because they fall under /usr. So, mounting your partitions in a very restrictive way is a double-edged sword. It limits what the hacker can do on your system, but it makes software installs and kernel upgrades more difficult (or impossible...if the partitions are still mounted in a restrictive way).

Given that, if you want to add new software or upgrade the kernel & ports tree source code, you'll need to

a. Change the partition's mounting in /etc/fstab back to their original values by copying your /etc/fstab.original file to /etc/fstab.
b. Bump the kernel security level back down to "1" by setting the kern_securelevel paramater in your /etc/rc.conf file, and then
c. Reboot the machine
d. Update your sources with cvsup, then make buildworld, make kernel, and make installworld

Then when you're done upgrading, recompiling, and installing, do the steps in reverse:

a. Change the partition's mounting in /etc/fstab to their restrictive values by copying your /etc/fstab.restrictive file to /etc/fstab.
b. Bump the kernel security level back up to "2" by setting the kern_securelevel paramater in your /etc/rc.conf file, and then
c. Reboot the machine

This may sound like a pain...I know. But this is your firewall, not a desktop workstation. This is the price you pay for a VERY, VERY secure machine. If you want an even more secure machine than this, then you can start setting the immutable flag on files in the filesystem by using the chflags command with the schg flag...but that's a whole separate howto in-and-of-itself. For now, though, you shouldn't need to manipulate immutable flags.

Reboot the machine so we can finish the job...

```
[root@numa /etc]# shutdown -r now
```

17. If the system doesn't reboot, it means that you probably made an error in the kernel configuration file...possibly setting the wrong type of CPU. DON'T PANIC. We can still boot the machine so that you can fix the error. To boot into the original version of the kernel, following the steps, below:

    A. Reboot the machine (power off, then on)

    B. When you reboot the machine and get to the part that says:

    ```
    Hit [Enter] to boot immediately, or any other key for command
    prompt.
    Booting [kernel] in 9 seconds...
    ```

    Hit the [space bar] (anything except the "enter" key), and you'll get to an "ok" prompt.

    C. Type in the following commands (at the "ok" prompt) and you'll boot the original kernel.

    ```
    ok unload kernel
    ok load kernel.old
    /kernel.old text=0xdf...bunch of stuff on the line...
    ok boot
    ```

    D. After the old kernel boots, then modify your kernel configuration file and fix whatever was causing the problems, recompile & install, and then reboot and continue with the next step.

18. As the system is rebooting, go through the kernel reconfiguration process if you need to configure your 2nd generic NE2000-compatible ISA Ethernet Card. Again, you'll only need to do this step if you have a 2nd NE2000-compatible ISA Ethernet card to configure. If you're using PCI network cards, you won't have to do this step.

    A. To do this, interrupt the boot process by hitting the [space bar] when you see the following:

    ```
    Hit [Enter] to boot immediately, or any other key for command
    prompt.
    Booting [kernel] in 9 seconds...
    ```

    B. Then, at the '>' prompt, type 'boot -c' and hit [Enter]. This will boot into the Kernel Configuration Utility

    C. Then, at the next '>' prompt (usually "'config>") type 'v' and hit [Enter] to start the Kernel Configuration Utility in visual mode. This will drop you off at the step 7 of the first section of this document (Installing FreeBSD-STABLE). This time, you'll see support for your 2nd Ethernet card, ed1. Configure the IRQ and Port, then save and continue booting

19. After the system comes back up, you'll want to re-generate the Tripwire database. Since you updated the kernal and all of the system binaries, the Tripwire database signatures of those files is out of date. If you don't update the Tripwire database, Tripwire will find thousands of "changes" to the system binaries when it runs for the first time at 4AM in the morning. To update the database so that it has signatures for the newest kernel & system binaries, etc, just type the following command:

    ```
    [root@numa /root]# tripwire --init --cfgfile /etc/tripwire/tw.cfg
    ```

    After you do this...you should have a completely working firewall...enjoy!