Linux Home Networking III

Beyond The Home

Chapter 1

Network Based Linux Installation

= =

In This Chapter

= =

RedHat Linux allows you to do operating system installations via a network connection. The procedure is fairly simple:

- Connect the new server (installation client) to the same network as the server with the pre-loaded installation files (installation server).
- Boot the installation client from a specially created floppy disk
- Enter your preferred installation method (FTP, HTTP, NFS) and the needed network parameters to do this
- The installation procedure will then continue with the more familiar RedHat Linux installation screens. Enter your selections and then complete the installation.
- This chapter will briefly explain how to set this up using all three methods using a single installation server (bigboy) with an IP address of 192.168.1.100.

Setting Up The Installation Server

Copy the Files

- In this example you can create a directory called **/data/RedHat9**
- Create another two directories under this one **/data/RedHat9/RedHat** and **/data/RedHat9/ISO**

HTTP & FTP Preparation

- Copy all the contents of the **/RedHat** directory of each of the installation CDs to the **/data/RedHat9/RedHat** directory. This will require about 2GB of space.

NFS Preparation

- Create ISO images of the installation CDs and place them in the **/data/RedHat9/ISO** directory. This will require about 2GB of space too.

First CD

```
[root@bigboy ISO]# cd /data/RedHat9/ISO
[root@bigboy ISO]# mount /mnt/cdrom
[root@bigboy ISO]# mkisofs -J -r -T -o shrike-i386-disc1.iso
/mnt/cdrom
Using RELEA000.HTM;1 for  /RELEASE-NOTES-de.html (RELEASE-
NOTES-es.html)
Using RELEA001.HTM;1 for  /RELEASE-NOTES-es.html (RELEASE-
NOTES-fr.html)
Using RELEA002.HTM;1 for  /RELEASE-NOTES-fr.html (RELEASE-
NOTES-it.html)
...
...
...
 95.37% done, estimate finish Tue Oct 21 08:46:18 2003
 96.88% done, estimate finish Tue Oct 21 08:46:16 2003
 98.39% done, estimate finish Tue Oct 21 08:46:15 2003
 99.91% done, estimate finish Tue Oct 21 08:46:14 2003
Total translation table size: 41696
Total rockridge attributes bytes: 64976
Total directory bytes: 110592
Path table size(bytes): 248
Max brk space used 80024
330304 extents written (645 Mb)
[root@bigboy ISO]# eject cdrom
```

Second CD

```
[root@bigboy ISO]# mount /mnt/cdrom
[root@bigboy ISO]# mkisofs -J -r -T -o shrike-i386-disc2.iso
/mnt/cdrom
[root@bigboy ISO]# eject cdrom
```

Third CD

```
[root@bigboy ISO]# mount /mnt/cdrom
[root@bigboy ISO]# mkisofs -J -r -T -o shrike-i386-disc3.iso
/mnt/cdrom
[root@bigboy ISO]# eject cdrom
```

Setup Your Webserver

- You will now have to setup Apache to give the file listings of your **/data/RedHat9/RedHat** and **/data/RedHat9/ISO** by pointing your browser to the URL http://192.168.1.200/RedHat9/RedHat/ or http://192.168.1.200/RedHat9/ISO/ respectively. A sample **httpd.conf** configuration is below.

```
NameVirtualHost 192.168.1.200

#
# For HTTP Installations
#
<VirtualHost 172.16.1.200>
     Alias /RedHat9 /data/RedHat9
     DocumentRoot /data/RedHat9/
</VirtualHost>

<Directory /data/RedHat9>
   Options +Indexes
   AllowOverride AuthConfig
   order allow,deny
   allow from all
</Directory>
```

- Remember to restart Apache to make these settings take effect

Setup Your FTP Server

- You'll also have to set up your VSFTP server to make incoming anonymous FTP connections log in to the /data/RedHat9 directory by default. Here is a sample snippet of the **vsftpd.conf** file

```
#
# Anonymous FTP Root Directory
#
anon_root=/data/RedHat9/
#
```

- Remember to restart VSFTP to make these settings take effect

Create A Special FTP User

- You can also create a special user for non anonymous FTP installations with its home directory as "/". You must also make sure that the user has read access to the **/data/RedHat9** directory. An example is below.

```
[root@bigboy tmp]# useradd -g users ftpinstall
[root@bigboy tmp]# passwd ftpinstall
Changing password for user ftpinstall.
New password:
Retype new password:
```

```
passwd: all authentication tokens updated successfully.
[root@bigboy tmp]#
[root@bigbot tmp]# usermod -d / ftpinstall
[root@bigbot tmp]#
```

Setup Your NFS Server

- Create a **/etc/exports** file with the following entry in it. You must use tabs, not spaces between the entries

```
/data/RedHat9           *(ro,sync)
```

- Run the **exportfs** command to add this directory to the NFS database of network available directories. You should also add this command to your **/etc/rc.local** file so that this is repeated after every reboot.

```
[root@bigboy tmp]# exportfs -ra
```

- Remember to restart NFS to make these settings take effect. You will also have to have the **/etc/init/portmap** daemon running as well.
- The installation client must have a matching pair of forward and reverse DNS entries on your DNS server. In other words, a DNS lookup on the IP address of the installation client must return a server name that will map back to the original IP address when a DNS lookup is done on that same server name.

```
[root@bigboy tmp]# host 192.168.1.96
96.1.168.192.in-addr.arpa domain name pointer 192-168-1-96.my-site.com.
[root@bigboy tmp]#

[root@bigboy tmp]# host 192-168-1-96.my-site.com
192-168-1-96.my-site.com has address 192.168.1.96
[root@bigboy tmp]#
```

This may mean that you will have to create entries for all your DHCP IP addresses if you choose to use a DHCP method of assigning IP addresses during installation.

Configure Your DHCP Server

During the installation procedure, the installation client will prompt you for the IP address it should use for the installation process. I recommend selecting the option that makes the Installation Client get its address via DHCP. This will automate the installation more and will therefore make it faster. It will also reduce the possibility of human error.

Setting up the Installation Server as a DHCP server is fairly straight forward and can be found in the Linux Home Networking guide.

Creating Boot Diskettes

You will have to create a set of two companion floppy diskettes, a boot diskette and a network driver diskette.

The first step is to insert the first Linux installation CD in your CD ROM drive and the blank boot diskette into your floppy drive. The next steps depend on whether you are creating the diskettes using a Windows or a Linux box.

Using Windows

- Execute the **rawrite.exe** file in the CDROM's **\dosutils** directory. The program will then prompt you for a filename. Enter the name of the boot image file **bootdisk.img** in the CDROM's **\images** directory

```
Enter disk image source file name: \images\bootdisk.img
Enter target diskette drive: a:
Please insert a formatted diskette into drive A: and press -
ENTER- :
```

- Remove the boot diskette and insert the blank network driver diskette into your floppy drive.
- Execute the rawrite.exe once again. Enter the name of the network driver image file **drvnet.img** in the CDROM's **\images** directory

```
Enter disk image source file name: \images\drvnet.img
Enter target diskette drive: a:
Please insert a formatted diskette into drive A: and press -
ENTER- :
```

- Remove the network driver diskette from your floppy drive.

Using Linux

- Mount the CDROM and then use the **dd** command to do a raw block copy of the boot image file **bootdisk.img** in the CDROM's **/images** directory directly to the floppy diskette.

```
[root@bigboy tmp]# mount /mnt/cdrom
[root@bigboy tmp]# dd if=/mnt/cdrom/images/bootdisk.img of=/dev/fd0
2880+0 records in
2880+0 records out
[root@bigboy tmp]#
```

- Remove the boot diskette and insert the blank network driver diskette into your floppy drive.

- Run the **dd** command again to copy the network driver image file **drvnet.img** in the CDROM's **/images** directory to the diskette

```
[root@bigboy tmp]# dd if=/mnt/cdrom/images/drvnet.img
of=/dev/fd0
2880+0 records in
2880+0 records out
[root@bigboy tmp]#
```

- Remove the network driver diskette from your floppy drive and eject the CDROM

```
[root@bigboy tmp]# eject cdrom
[root@bigboy tmp]#
```

The Network Installation

- Boot your system using the boot diskette
- Go through the usual steps until it prompts for the "Installation Method". You will see a number of choices

```
Local CDROM
Hard Drive
NFS Image
FTP
HTTP
```

- Select the network option of your choice (NFS, FTP, HTTP)
- You will then get a "No Driver Found" screen at which point you should remove the boot diskette and replace it with the driver diskette.
- Select the "Use a Driver Disk" option, choose the floppy device "**fd0**" as the source of the files and then proceed to the "Networking Device" menu
- Select the Ethernet device to which the installation client is connected to the installation server network. This would most likely be interface "**eth0"**.
- Select "DHCP" in the following "Configure TCP/IP" screen. This will make the Installation client use DHCP during the installation.

If You Selected The NFS Method

- You will now reach the "NFS setup" menu.
- Enter the IP address of the installation server as the "NFS Server Name". The "RedHat directory" will be "**/data/RedHat9/ISO**".
- The following menus will be the usual RedHat GUI installation screens.

If You Selected The HTTP Method

- You will now reach the “HTTP Setup” menu.
- Enter the IP address of the installation server. The “RedHat directory” will be “/RedHat9”.
- The following menus will be text based versions of the usual RedHat installation screens.

If You Selected The FTP Method

- You will now reach the “FTP Setup” menu.
- Enter the IP address of the installation server as the “FTP Site Name”.

Not Selecting The Non-Anonymous FTP Box

- The “RedHat directory” will be “/”.
- The following menus will be text based versions of the usual RedHat installation screens.

Selecting The Non-Anonymous FTP Box

- The “RedHat directory” will be “/data/RedHat9”.
- Enter the username and password of your special FTP user account
- The following menus will be text based versions of the usual RedHat installation screens.

Troubleshooting The Network Installation

You can do some basic troubleshooting by accessing the various installation status screens available.

- The installation logs can always be viewed by hitting <CTRL-ALT-F3>
- Kernel messages can be seen by hitting <CTRL-ALT-F4>
- Access to a limited BASH shell Kernel can be gained by hitting <CTRL-ALT-F2>
- You can return to the main installation screen at any time by hitting <CTRL-ALT-F1>

Automating Installation With Redhat Kickstart

RedHat Linux saves all the parameters you used during installation in the **/root/anaconda-ks.cfg** kickstart configuration file. You can use this file to create an automated installation of a duplicate system which can be useful if you have a large number of servers to install.

Creating A New Kickstart Configuration File

You can create your own brand new kickstart configuration file by using the "**ksconfig**" command from a GUI console. It will bring up a menu from which you can select all your installation options. When finished, you save the configuration with the filename of your choice.

Note: During the Kickstart process you will be prompted for any of the options you didn't specify with "**ksconfig**". Prompting could be useful if you want to change things like IP addressing and disk partitioning with each installation.

Note: Do not change the order of the entries in the kickstart configuration file.

Once you have created the kickstart configuration file you can copy it to:

- A directory on the server which has your network installation files
- A boot diskette
- A regular Linux diskette

Copying The File To A Linux Boot Diskette

The file should have the name "**ks.cfg**" and should be placed in the top directory of the diskette. As Linux boot diskettes are in MSDOS format you will have to use the "**mcopy**" command to copy the file:

```
[root@bigboy tmp]# mcopy ks.cfg a:ks.cfg
```

The kickstart process will fail to read your configuration file if you just use the regular "**cp**" command.

Copying the file to a Regular Linux diskette

This is sometimes needed when you install from CDROMs and need to get the kickstart configuration file from another device. As in the case of the Linux Boot diskette method, the file should be named "**ks.cfg**".

```
[root@bigboy tmp]# cp ks.cfg /mnt/floppy/ks.cfg
```

Copying The File To Your Kickstart Server

Whether you are using NFS, HTTP or FTP for your network installation it is best to place your kickstart files in a subdirectory under the RedHat directory. The examples below assume the subdirectory is called **RedHat/kickstart**.

A Note About Using anaconda-ks.cfg

It is possible to use the **/root/anaconda-ks.cfg** file as a template for future installations. RedHat comments out the partitioning information in this file, so you will either have to uncomment it and then make your partitioning modifications or be prepared to be prompted for your portioning information.

Booting With Your Kickstart Files

Once you have booted from your boot device, a floppy or CDROM, you'll need to use the following commands at the lilo boot: prompt to continue with the installation.

Floopy Boot, ks.cfg On Boot Floppy

```
boot: linux ks=floppy
```

CDROM Boot, ks.cfg On Floppy

```
boot: linux ks=hd:fd0/ks.cfg
```

The HTTP Server Has ks.cfg

```
boot: linux ks=http:192.168.1.100:/kickstart/ks.cfg
```

The NFS Server Has ks.cfg

Explicitly Stating the Configuration Filename

You can configure kickstart to use a particular file under the /data/RedHat9/ directory that our NFS configuration on the installation server makes available.

```
boot: linux ks=nfs:192.168.1.100:/kickstart/ks.cfg
```

Implicit Configuration Filename

```
boot: linux ks
```

Kickstart will use NFS to get the configuration file, but the file name used will depend on the settings of the DHCP server used on the kickstart network. In all the cases below, the NFS server IP address is assumed to be, 192.168.1.100. The "**next-server**" isn't necessary if the DHCP and NFS servers are the same box.

- If the network's "filename" setting in the dhcpd.conf file begins with a "/", kickstart will interpret this as the full path to the configuration file.

```
filename "/usr/new-machine/kickstart/ks.cfg";
next-server 192.168.1.100
```

- If the network's "filename" setting in the dhcpd.conf file does not begin with a "/" kickstart will interpret this as a file in the **/kickstart** directory.

```
filename "ks.cfg";
next-server 192.168.1.100
```

- If there is no filename specified in the **dhcpd.conf** file, kickstart will expect the file name to be **/kickstart/ipaddress-kickstart**. For example:

```
192.168.1.10-kickstart
```

Chapter 2

Linux Software RAID

= =

In This Chapter

= =

The main goals of using Redundant Arrays of Inexpensive Disks (RAID) is to either improve disk data performance and/or provide data redundancy.

RAID can be handled either by the operating system software or it may be implemented via a purpose built RAID disk controller card without having to configure the operating system at all. This chapter will attempt to explain how to configure the software RAID schemes supported by RedHat Linux 9 and newer.

For the sake of simplicity, this chapter focuses on using RAID for partitions that include neither the /boot or the root (/) filesystems.

RAID Types

Whether hardware or software based, RAID can be configured using a variety of standards. The most popular ones are listed below.

Linear Mode RAID

- In this scheme, the RAID controller views the RAID set as a “chain” of disks. Data is only written to the next device in the chain after the previous one has been filled.
- The aim of Linear RAID is to accommodate large filesystems spread over multiple devices with no data redundancy. A drive failure will corrupt your data.
- Linear mode RAID is supported by RedHat Linux.

RAID 0

- The RAID controller tries to evenly distribute data across all disks in the RAID set. A good analogy would be to envisage a file as if it were a book. In RAID 0, the system will try to sequentially place each “page” of the file on a different disk. In reality, the pages are called “chunks” whose size is determined when you initially configure RAID. So for example, in a RAID set of 3 disks, the first “chunk” of data in a file will be placed on disk 1, the second chunk will be on disk 2, the third chunk will on disk 3, the fourth chunk will then be placed on disk 1, etc. It is for this reason that RAID 0 is often called “striping”.
- Like Linear RAID, RAID 0 aims to accommodate large file systems spread over multiple devices with no data redundancy. The advantage of RAID 0 is data access speed. A file that is spread over three disks can be read three times as fast.
- RAID 0 can accommodate disks of unequal sizes. When RAID runs out of “striping space” on the smallest device, it then continues the striping using the available space on the remaining drives. When this occurs, the data access speed is lower for this portion of data as the total number of RAID drives available is reduced. It is for this reason that RAID 0 is best used with equal sized drives.
- RAID 0 is supported by RedHat Linux.

RAID 1

- With RAID 1, data is cloned on a duplicate disk. This RAID method is therefore frequently called “disk mirroring”.
- When one of the disks in the RAID set fails, the other one continues to function. When the failed disk is replaced, the data is automatically cloned to the new disk from the surviving disk. RAID 1 also offers the possibility of using a “hot standby” spare disk which will be automatically cloned in the event of a disk failure on any of the primary RAID devices.
- RAID 1 offers data redundancy, without the speed advantages of RAID 0. A disadvantage of software based RAID 1 is that the server has to send data twice to be written to each of the mirror disks. This can saturate data busses and CPU utilization. With a hardware based solution, the server CPU sends the data to the RAID disk controller once, and the disk controller then duplicates the data to the mirror disks. This makes fact often makes RAID capable disk controllers the preferred solution when implementing RAID 1.
- A limitation of RAID 1 is that the total RAID size in Gigabytes is equal to that of the smallest disk in the RAID set. Unlike RAID 0, the extra space on the larger device isn’t used.
- RAID 1 is supported by RedHat Linux.

RAID 4

- Let’s return to the book analogy from our description of RAID 0. RAID 4 operates similarly, but inserts a special error correcting or parity “page” on an additional disk dedicated to this purpose.

- RAID 4 requires at least three disks in the RAID set and can only survive the loss of a single drive. When this occurs, the data in can be recreated “on the fly” with the aid of the information on the RAID set’s parity disk. When the failed disk is replaced, is repopulated with the “lost data” with the help of the parity disk’s information.
- RAID 4 combines the goal of high speed provided by RAID 0 with the redundancy goal of RAID 0. Its major disadvantage is that the data is striped, but the parity information is not. In other words, any data written to the any section of the data portion of the RAID set must be followed by an update of the parity disk. The parity disk can therefore act as a bottleneck. It is for this reason that RAID 4 isn’t used very frequently.
- RAID 4 is not supported by RedHat Linux.

RAID 5

- RAID 5 improves on RAID 4 by striping the parity data between all the disks in the RAID set, This avoids the parity disk bottleneck while maintaining many of the speed features of RAID 0 and the redundancy of RAID 1. Like RAID 4, RAID 5 can only survive the loss of a single disk.
- RAID 5 is supported by RedHat Linux.

Before You Start

There are some basic guidelines you may want to follow when setting up RAID.

IDE Drives

Most home & SOHO systems will probably use IDE disks They do have some limitations.

- The total length of an IDE cable can only be a few feet long which generally limits their use to small home systems.
- IDE drives do not “hot swapping”. You cannot replace them while your system is running.
- Only two devices can be attached per controller.
- The performance of the IDE bus can be degraded by the presence of a second device on the cable.
- The failure of one drive on an IDE bus often causes the malfunctioning of the second device. This can be fatal if you have two IDE drives of the same RAID set attached to the same cable.

It is for these reasons that it is recommended to use only one IDE drive per controller when using RAID, especially in a corporate environment. In a home or SOHO setting, IDE based software RAID may be adequate.

SCSI Drives

SCSI hard disks have a number of features that make them more attractive for RAID use.

- SCSI controllers are more tolerant of disk failures. The failure of a single drive is less likely to disrupt the remaining drives on the bus.
- SCSI cables can be several meters long, making them suitable for data center applications.
- Much more than two devices may be connected to a SCSI cable bus.
- Some models of SCSI devices support “hot swapping” which allows you to replace them while the system is running.

However SCSI drives tend to be more expensive than IDE drives which may make them less attractive for home use.

Should I Software RAID Partitions Or Entire Disks?

It is generally a not a good idea to share RAID configured partitions with non RAID partitions. The reason for this is obvious as a disk failure could still incapacitate a system.

If you decide to use RAID, all the partitions on each RAID disk should be part of a RAID set. Many people simplify this problem by filling each disk of a RAID set with only one partition.

Configuring Software RAID

RAID Partitioning

You will first need to identify two or more partitions, each on a separate disk. If you are doing RAID 0 or RAID 5, the partitions should be of approximately the same size, as in this scenario, RAID will limit the extent of data access on each partition to an area no larger than that of the smallest partition in the RAID set.

In this example we’ll be configuring RAID 5 using a system with three pre-partitioned hard disks. The partitions to be used will be:

- /dev/hde1
- /dev/hdf2
- /dev/hdg1

Start FDISK

- You have to change each RAID partition used to be of type FD (Linux raid autodetect). This can be done using **fdisk**. Here is an example using /dev/hde1

```
[root@bigboy updates]# fdisk /dev/hde

The number of cylinders for this disk is set to 8355.
There is nothing wrong with that, but this is larger than
1024,
and could in certain setups cause problems with:
1) software that runs at boot time (e.g., old versions of
LILO)
2) booting and partitioning software from other OSs
   (e.g., DOS FDISK, OS/2 FDISK)

Command (m for help):
```

Use FDISK Help

- We now use the **fdisk** "m" command to get some help

```
Command (m for help): m

Command action
   a   toggle a bootable flag
   b   edit bsd disklabel
   c   toggle the dos compatibility flag
   d   delete a partition
   l   list known partition types
   m   print this menu
   n   add a new partition
   o   create a new empty DOS partition table
   p   print the partition table
   q   quit without saving changes
   s   create a new empty Sun disklabel
   t   change a partition's system id
   u   change display/entry units
   v   verify the partition table
   w   write table to disk and exit
   x   extra functionality (experts only)

Command (m for help):
```

Set The ID Type To FD

- Partition **/dev/hde1** is the 1st partition on disk **/dev/hde**. We now modify its "type" using the "t" command and then specifying the partition number and type code. We also use the "L" command to get a full listing of ID types in case we forget.

```
Command (m for help): t
Partition number (1-5): 1
Hex code (type L to list codes): L
```

```
 0  Empty           1c  Hidden Win95 FA 70  DiskSecure Mult bb  Boot Wizard hid
 1  FAT12           1e  Hidden Win95 FA 75  PC/IX           be  Solaris boot
 2  XENIX root      24  NEC DOS         80  Old Minix       c1  DRDOS/sec (FAT-
 3  XENIX usr       39  Plan 9          81  Minix / old Lin c4  DRDOS/sec (FAT-
 4  FAT16 <32M      3c  PartitionMagic  82  Linux swap      c6  DRDOS/sec (FAT-
 5  Extended        40  Venix 80286     83  Linux           c7  Syrinx
 6  FAT16           41  PPC PReP Boot   84  OS/2 hidden C:  da  Non-FS data
 7  HPFS/NTFS       42  SFS             85  Linux extended  db  CP/M / CTOS / .
 8  AIX             4d  QNX4.x          86  NTFS volume set de  Dell Utility
 9  AIX bootable    4e  QNX4.x 2nd part 87  NTFS volume set df  BootIt
 a  OS/2 Boot Manag 4f  QNX4.x 3rd part 8e  Linux LVM       e1  DOS access
 b  Win95 FAT32     50  OnTrack DM      93  Amoeba          e3  DOS R/O
 c  Win95 FAT32 (LB 51  OnTrack DM6 Aux 94  Amoeba BBT      e4  SpeedStor
 e  Win95 FAT16 (LB 52  CP/M            9f  BSD/OS          eb  BeOS fs
 f  Win95 Ext'd (LB 53  OnTrack DM6 Aux a0  IBM Thinkpad hi ee  EFI GPT
10  OPUS            54  OnTrackDM6      a5  FreeBSD         ef  EFI (FAT-12/16/
11  Hidden FAT12    55  EZ-Drive        a6  OpenBSD         f0  Linux/PA-RISC b
12  Compaq diagnost 56  Golden Bow      a7  NeXTSTEP        f1  SpeedStor
14  Hidden FAT16 <3 5c  Priam Edisk     a8  Darwin UFS      f4  SpeedStor
16  Hidden FAT16    61  SpeedStor       a9  NetBSD          f2  DOS secondary
17  Hidden HPFS/NTF 63  GNU HURD or Sys ab  Darwin boot     fd  Linux raid auto
18  AST SmartSleep  64  Novell Netware  b7  BSDI fs         fe  LANstep
1b  Hidden Win95 FA 65  Novell Netware  b8  BSDI swap       ff  BBT
Hex code (type L to list codes): fd
Changed system type of partition 1 to fd (Linux raid autodetect)

Command (m for help):
```

Make Sure The Change Occurred

- Use the “p” command to get the new proposed partition table

```
Command (m for help): p

Disk /dev/hde: 4311 MB, 4311982080 bytes
16 heads, 63 sectors/track, 8355 cylinders
Units = cylinders of 1008 * 512 = 516096 bytes

   Device Boot    Start       End    Blocks   Id  System
/dev/hde1             1      4088   2060320+  fd  Linux raid autodetect
/dev/hde2          4089      5713    819000   83  Linux
/dev/hde3          5714      6607    450576   83  Linux
/dev/hde4          6608      8355    880992    5  Extended
/dev/hde5          6608      7500    450040+  83  Linux
/dev/hde6          7501      8355    430888+  83  Linux

Command (m for help):
```

Save The Changes

- Use the “w” command to permanently save the changes to disk **/dev/hde.**

```
Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.

WARNING: Re-reading the partition table failed with error 16: Device or
resource busy.
The kernel still uses the old table.
```

```
The new table will be used at the next reboot.
Syncing disks.
[root@bigboy updates]#
```

- The error above will occur if any of the other partitions on the disk is mounted.

Repeat For The Other Partitions

- For the sake of brevity, I won't show the process to do this, but the steps for changing the IDs for **/dev/hdf2** and **/dev/hdg1** are very similar.

Edit The RAID Configuration File

The linux RAID configuration file is **/etc/raidtab**. A good template file for **/etc/raidtab** can be found in the man pages, simply issue the command "man raidtab".

General Guidelines

- When configuring RAID 5 a "parity-algorithm" setting must be used.
- The "raid-disk" parameters for each partition in the **/etc/raidtab file** are numbered starting at "0".
- The **/etc/raidtab** "persistent-superblock" must be set to "1" in order for the RAID autodetect feature (partition type FD) to work.

In our example:

- We configure RAID 5 on using each of the desired partitions on the 3 disks.
- All other RAID levels use a "persistent-superblock" setting of "0".
- The set of 3 RAID disks will be called **/dev/md0**.

```
#
# sample raiddev configuration file
# 'old' RAID0 array created with mdtools.
#
raiddev /dev/md0
raid-level              5
nr-raid-disks           3
persistent-superblock   1
chunk-size              32
parity-algorithm        left-symmetric
device                  /dev/hde1
raid-disk               0
device                  /dev/hdf2
raid-disk               1
device                  /dev/hdg1
raid-disk               2
```

Create the RAID Set

The **mkraid** command creates the RAID set by reading the **/etc/raidtab** file. In this case we want to create the logical RAID device /dev/md0

```
[root@bigboy root]# mkraid /dev/md0
handling MD device /dev/md0
analyzing super-block
[root@bigboy root]#
```

Format The New RAID Set

Your new RAID device will now have to be formatted. In the example below:

- We use the "**-j**" qualifier to ensure that a journaling file systems is created.
- A block size of 4KB (4096 bytes) is used with each chunk being comprised of 8 blocks. It is very important that the "chunk-size" parameter in the **/etc/raidtab** file match the value of the block size multiplied by the stride value in the command below. **Note:** If the values don't match, then you will get parity errors.

```
[root@bigboy root]# mke2fs -j -b 4096 -R stride=8 /dev/md0
mke2fs 1.32 (09-Nov-2002)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
516096 inodes, 1030160 blocks
51508 blocks (5.00%) reserved for the super user
First data block=0
32 block groups
32768 blocks per group, 32768 fragments per group
16128 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736

Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 26 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
[root@bigboy root]#
```

Load The RAID Driver For The New RAID Set

The next step is make the Linux operating system fully aware of the RAID set by loading the driver for the new RAID set using the **raidstart** command.

```
root@bigboy root]# raidstart /dev/md0
[root@bigboy root]#
```

Create A Mount Point For The RAID Set

The next step is to create a mount point for /dev/md0. In this case we'll create one called /mnt/raid

```
[root@bigboy mnt]# mkdir /mnt/raid
```

Edit The /etc/fstab File

We'll now add an entry for the /dev/md0 device. Here is an example of a line that could be used:

```
/dev/md0        /mnt/raid      ext3     defaults     1 2
```

Note: It is very important that you DO NOT use labels in the **/etc/fstab** file for RAID devices, just use the real device name such as "/dev/md0". On startup, the **/etc/rc.d/rc.sysinit** script checks the **/etc/fstab** file for device entries that match RAID set names in the **/etc/raidtab** file. It will not automatically start the RAID set driver for the RAID set if it doesn't find a match. Device mounting then occurs later on in the boot process. Mounting a RAID device that doesn't have a loaded driver can corrupt your data giving the error below.

```
Starting up RAID devices: md0(skipped)
Checking filesystems
/raiddata: Superblock has a bad ext3 journal(inode8)
CLEARED.
***journal has been deleted - file system is now ext 2 only***

/raiddata: The filesystem size (according to the superblock) is
2688072 blocks.
The physical size of the device is 8960245 blocks.
Either the superblock or the partition table is likely to be
corrupt!
/boot: clean, 41/26104 files, 12755/104391 blocks

/raiddata: UNEXPECTED INCONSISTENCY; Run fsck manually (ie without
-a or -p options).
```

Start The New RAID Set's Driver

This is done using the **raidstart** command.

```
[root@bigboy root]# raidstart /dev/md0
```

Mount The New RAID Set

The mount command can now be used to mount the RAID set.

Using the automount feature

The **mount** command's "-a" flag will cause Linux to mount all the devices in the /etc/fstab file that have automounting enabled (default) and that are also not already mounted.

```
[root@bigboy root]# mount –a
```

Manually

You can also mount the device manually.

```
[root@bigboy root]# mount /dev/md0 /mnt/raid
```

Check The Status Of The New RAID

The **/proc/mdstat** file provides the current status of all the devices. When the raid driver is stopped, the file has very little information as seen below

```
[root@bigboy root]# raidstop /dev/md0
[root@bigboy root]# cat /proc/mdstat
Personalities : [raid5]
read_ahead 1024 sectors
unused devices: <none>
[root@bigboy root]#
```

More information, including the partitions of the RAID set, is provided once the driver is loaded using the **raidstart** command.

```
[root@bigboy root]# raidstart /dev/md0
[root@bigboy root]# cat /proc/mdstat
Personalities : [raid5]
read_ahead 1024 sectors
md0 : active raid5 hdg1[2] hde1[1] hdf2[0]
      4120448 blocks level 5, 32k chunk, algorithm 3 [3/3] [UUU]

unused devices: <none>
[root@bigboy root]#
```

Chapter 3

Expanding Linux Partitions With LVM

= =

In This Chapter

= =

The RedHat Linux Logical Volume Manager (LVM) allows you to resize your partitions without having to modify the partition tables on your hard disk. This is most useful when you find yourself running out of space on a filesystem and want to expand into a new disk partition versus migrating the filesystem to a new disk.

LVM Terminologies

Physical Volume

A physical Volume (PV) is another name for a regular physical disk partition that is used or will be used by LVM.

Logical Volume

Any number of partitions (PVs) on different disk drives can be lumped together into a logical volume (LV). Under LVM, logical volumes are analogous to a “virtual disk drive”.

Volume Groups

Logical volumes must then be subdivided into volume groups. Each volume group can be individually formatted as if it were a regular Linux partition. A volume group is therefore like a “virtual partition” on your “virtual disk drive”.

This may seem complicated but it allows you to create new virtual partitions with sizes you can change from groups of real disk partitions whose sizes you probably cannot change. Another advantage of LVM is that this can all be done without disturbing other partitions on your hard disks.

Physical Extent

Real disk partitions are divided into chunks of data called physical extents (PEs) when you add them to a logical volume. PEs are important as you usually have to specify the size of your volume group not in Gigabytes, but as a number of physical extents.

Configuring LVM Devices

Example Background

In this example, our **/home** filesystem which resides on **/dev/hde5** has become too full. We've just added a new hard drive **/dev/hdf** with 50% of the capacity of **/dev/hde5** into which we want to expand **/home**. The single partition of **/dev/hdf** is **/dev/hdf1**.

Preparation

Backup Your Data

Use the **tar** command or some other method to backup your data in **/home**. The LVM process will destroy the data on all physical volumes.

Unmount your /home filesystem

As **/home** stores most user's data, you'll need to ensure they are all logged off by:

- Logging into the system video console
- Going into single user mode

```
[root@bigboy root]# init 1
```

- And finally unmounting the file system

```
[root@bigboy root]# umount /home
```

Determine The Partition Types

You have to change each LVM partition used to be of type 8e (Linux LVM). You can test this with the "**fdisk –l**" command. Here is an example using **/dev/hde** showing that our target partitions are of the incorrect type**.**

```
[root@bigboy root]# fdisk -l /dev/hde

Disk /dev/hde: 4311 MB, 4311982080 bytes
16 heads, 63 sectors/track, 8355 cylinders
Units = cylinders of 1008 * 512 = 516096 bytes
```

```
Device Boot        Start      End    Blocks   Id  System
/dev/hde1             1      4088   2060320+  fd  Linux raid
autodetect
/dev/hde2          4089      5713    819000   83  Linux
/dev/hde3          5714      6607    450576   83  Linux
/dev/hde4          6608      8355    880992    5  Extended
/dev/hde5          6608      7500    450040+  83  Linux
/dev/hdf1          7501      8355    430888+  83  Linux
[root@bigboy root]#
```

Start FDISK

- You can change the partition type using **fdisk** too. Here is an example in which we change **/dev/hde5** and **/dev/hdf1.**

```
[root@bigboy root]# fdisk /dev/hde

The number of cylinders for this disk is set to 8355.
There is nothing wrong with that, but this is larger than
1024,
and could in certain setups cause problems with:
1) software that runs at boot time (e.g., old versions of
LILO)
2) booting and partitioning software from other OSs
   (e.g., DOS FDISK, OS/2 FDISK)

Command (m for help):
```

Set The ID Type To 8e

- Partitions **/dev/hde5** and **/dev/hdf1** are the 5th and 6th partitions on disk **/dev/hde**. We now modify their "type" using the "t" command and then specifying the partition number and type code. We can also use the "L" command to get a full listing of ID types in case we forget.

```
Command (m for help): t
Partition number (1-6): 5
Hex code (type L to list codes): 8e
Changed system type of partition 5 to 8e (Linux LVM)

Command (m for help): t
Partition number (1-6): 6
Hex code (type L to list codes): 8e
Changed system type of partition 6 to 8e (Linux LVM)

Command (m for help):
```

Make Sure The Change Occurred

- Use the “p” command to get the new proposed partition table

```
Command (m for help): p

Disk /dev/hde: 4311 MB, 4311982080 bytes
16 heads, 63 sectors/track, 8355 cylinders
Units = cylinders of 1008 * 512 = 516096 bytes

Device Boot      Start         End      Blocks   Id  System
/dev/hde1            1        4088   2060320+  fd  Linux raid
autodetect
/dev/hde2         4089        5713    819000   83  Linux
/dev/hde3         5714        6607    450576   83  Linux
/dev/hde4         6608        8355    880992    5  Extended
/dev/hde5         6608        7500    450040+  8e  Linux LVM
/dev/hdf1         7501        8355    430888+  8e  Linux LVM

Command (m for help):
```

Save The Changes

- Use the “w” command to permanently save the changes to disk **/dev/hde.**

```
Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.

WARNING: Re-reading the partition table failed with error 16:
Device or resource busy.
The kernel still uses the old table.
The new table will be used at the next reboot.
Syncing disks.
[root@bigboy updates]#
```

- The error above will occur if any of the other partitions on the disk is mounted.

Run VGscan

The next step is to make Linux scan for any new LVM disk partitions and automatically create the LVM configuration files in the /etc directory. The **vgscan** command is used to do this.

```
[root@bigboy root]# vgscan
vgscan -- reading all physical volumes (this may take a while...)
vgscan -- "/etc/lvmtab" and "/etc/lvmtab.d" successfully created
vgscan -- WARNING: This program does not do a VGDA backup of your
volume group

[root@bigboy root]#
```

Define Each Physical Volume

You then initialize the target partitions with the **pvcreate** command. This will wipe out all the data on them in preparation for the next step. If you haven't backed up your data yet, do it now!

```
[root@bigboy root]# pvcreate /dev/hde5
pvcreate -- physical volume "/dev/hde5" successfully created

[root@bigboy root]# pvcreate /dev/hdf1
pvcreate -- physical volume "/dev/hdf1" successfully created

[root@bigboy root]#
```

Create A Volume Group For the PVs

- The **vgcreate** command is then used to combine the two physical volumes into a single unit called a "volume group". The LVM software effectively tricks the operating system into thinking the volume group is a new hard disk.
- In our example, we've decided to call the volume group "lvm-hde".

```
[root@bigboy root]# vgcreate lvm-hde /dev/hdf1 /dev/hde5
vgcreate -- INFO: using default physical extent size 4 MB
vgcreate -- INFO: maximum logical volume size is 255.99
Gigabyte
vgcreate -- doing automatic backup of volume group "lvm-hde"
vgcreate -- volume group "lvm-hde" successfully created and
activated

[root@bigboy root]#
```

Create A Logical Volume From The Volume Group

The next step is to partition the volume group into "logical volumes" with the **lvcreate** command. Like hard disks, which are divided into blocks of data, logical volumes are divided into units called physical extents (PEs).

You'll have to know the number of available PEs before creating the logical volume. This is done with the **vgdisplay** command.

```
[root@bigboy root]# vgdisplay lvm-hde
--- Volume group ---
VG Name               lvm-hde
VG Access             read/write
VG Status             available/resizable
VG #                  0
MAX LV                256
Cur LV                0
Open LV               0
```

```
MAX LV Size            255.99 GB
Max PV                 256
Cur PV                 2
Act PV                 2
VG Size                848 MB
PE Size                4 MB
Total PE               212
Alloc PE / Size        0 / 0
Free  PE / Size        212 / 848 MB
VG UUID                W7bgLB-lAFW-wtKi-wZET-jDJF-8VYD-snUaSZ


[root@bigboy root]#
```

Here you can see we have 212 PEs available as “free”. We can now use all 212 of them to create a logical volume named **lvm0** from volume group **lvm-hde**.

```
[root@bigboy root]# lvcreate -l 212 lvm-hde -n lvm0
lvcreate -- doing automatic backup of "lvm-hde"
lvcreate -- logical volume "/dev/lvm-hde/lvm0" successfully
created

[root@bigboy root]#
```

Format The Volume Group

Once the logical volume is created, we can now format it as if it were a regular partition.

```
[root@bigboy root]# mkfs -j  /dev/lvm-hde/lvm0
mke2fs 1.32 (09-Nov-2002)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
108640 inodes, 217088 blocks
10854 blocks (5.00%) reserved for the super user
First data block=0
7 block groups
32768 blocks per group, 32768 fragments per group
15520 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840

Writing inode tables: done
Creating journal (4096 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 38 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
[root@bigboy root]#
```

Mount The Volume Group

The next steps are more straightforward.

Label The Volume Group

- This is done with the e2label command. Here we label it "**/lvm0**"

```
[root@bigboy root]# e2label /dev/lvm-hde/lvm0 /lvm0
```

Create A Mount Point

- We can make this be **/mnt/lvm**.

```
[root@bigboy root]# mkdir /mnt/lvm
```

Update The /etc/fstab File

- In this snippet, we configure the newly labeled partition to be mounted on the **/mnt/lvm** mount point.

```
LABEL=/lvm0    /mnt/lvm            ext3    defaults        1 2
```

Mount The Volume

- The "**mount –a**" command reads the **/etc/fstab** file and mounts all the devices that haven't been mounted already. After mounting we test the volume by listing its directory contents.

```
[root@bigboy root]# mount -a
[root@bigboy root]# ls /mnt/lvm
lost+found
[root@bigboy root]#
```

Create A Logical Link For /home

You now need to create a logical link for **/home** that points to **/mnt/lvm**.

```
[root@bigboy root]# ln –s  /mnt/lvm  /home
```

Restore Your Data

You can now restore your backed up data to **/home**.

Get Out Of Single User Mode

Return to your original run state by using either the “**init 3**” or “**init 5**” commands.

Chapter 4

Managing Disk Usage With Quotas

= =

In This Chapter

= =

You may eventually need to restrict the amount of disk space used on each partition by each user or group of users. RedHat Linux allows this to occur by using its disk quota feature. The setup is fairly simple.

Setting Up Quotas

In our example, the family Linux server is running out of space in the **/home** filesystem due to a lot of MP3 downloads.

Enter Single User Mode

As we'll need to remount the **/home** filesystem it's best to ensure that no other users or processes are using it. This is best achieved by entering single user mode from the console. This may be unnecessary if you are certain that you're the only user on the system.

```
[root@bigboy tmp]# init 1
```

Edit Your /etc/fstab File

You have to alert Linux that quotas are enabled on the filesystem by editing the **/etc/fstab** file and modifying the options for the /home directory. You'll need to add the **usrquota** option. In case you forget the name, the **usrquota** option is mentioned in the **fstab** man pages.

Old fstab

```
LABEL=/home        /home           ext3    defaults            1 2
```

New fstab

```
LABEL=/home        /home           ext3    defaults,usrquota  1 2
```

Remount The Filesystem

Editing the **/etc/fstab** file isn't enough, Linux needs to reread the file to get its instructions for **/home**. This can be done using the **mount** command with the "**-o remount**" qualifier.

```
[root@bigboy tmp]# mount -o remount /home
[root@bigboy tmp]#
```

Create The Partition Quota Configuration Files

The topmost directory of the filesystem needs to have a **aquota.user** file (Defines quotas by user) and/or a **aquota.group** file (Defines quotas by user). The man page for "**quota**" lists them at the bottom.

In this case we'll just enable "per user " quotas.

```
[root@bigboy tmp]# touch /home/aquota.user
[root@bigboy tmp]# chmod 600 /home/aquota.user
[root@bigboy tmp]#
```

Make Linux Read The Quota Config File

This is done using the quotacheck command. You'll get an error the first time you enter the command as Linux will realize that the file wasn't created using one of the quota commands.

```
[root@bigboy tmp]# quotacheck -vagum
quotacheck: WARNING -  Quotafile /home/aquota.user was probably
truncated. Can't save quota settings...
quotacheck: Scanning /dev/hda3 [/home] done
quotacheck: Checked 185 directories and 926 files
[root@bigboy tmp]#
```

Edit The User's Quota Information

Now we need to edit the user's quota information. This is done with the **edquota** command which allows you to selectively edit a portion of the **aquota.user** file on a per user basis.

```
[root@bigboy tmp]# edquota -u mp3user
```

The command will invoke the **vi** editor which will allow you to edit a number of fields.

```
Disk quotas for user mp3user (uid 503):
  Filesystem  blocks       soft       hard     inodes     soft     hard
  /dev/hda3       24          0          0          7        0        0
```

Blocks: The amount of space in 1K blocks the user is currently using.

Inodes: The number of files the user is currently using.

Soft Limit: The maximum blocks/inodes a quota user may have on a partition. The role of a soft limit changes if grace periods are used. When this occurs, the user is only warned that their soft limit has been exceeded. When the grace period expires, the user is barred from using additional disk space or files. When set to zero, limits are disabled.

Hard Limit: The maximum blocks/inodes a quota user may have on a partition when a grace period is set. Users may exceed a soft limit, but they can never exceed their hard limit.

In the example below we limit user **mp3user** to a maximum of 5 MB of data storage on **/dev/hda3** (**/home**).

```
Disk quotas for user mp3user (uid 503):
  Filesystem  blocks       soft       hard     inodes     soft     hard
  /dev/hda3       24       5000          0          7        0        0
```

Get Out Of Single User Mode

Return to your original run state by using either the "**init 3**" or "**init 5**" commands.

Other Quota Topics

Editing Grace Periods

The "**edquota -t**" command sets the grace period for each filesystem. Like the **edquota** command, it invokes the **vi** editor.

The grace period is a time limit before the soft limit is enforced for a quota enabled file system. Time units of seconds, minutes, hours, days, weeks and months can be used. This is what you'll see with the command "**edquota -t**":

Note: There should be no spaces between the number and the unit of time measure. Therefore in this example, "**7days**" is correct and "**7 days**" is wrong.

```
[root@bigboy tmp]# edquota –t

Grace period before enforcing soft limits for users:
Time units may be: days, hours, minutes, or seconds
  Filesystem             Block grace period     Inode grace period
  /dev/hda3                     7days                  7days
```

Editing Group Quotas

Editing quotas on a per group basis can be done similarly with the "**edquota –g**" command.

Checking Quotas Regularly

Once Linux is aware of users who have exceeded their quotas, the operating system subsequently prevents these users from creating more files or using more disk space in the partition.

Linux doesn't check quota usage each time a file is opened, you have to force it to process the **aquota.user** and **aquota.group** files periodically with the **quotacheck** command.

You can setup a **cron** job to run a script similar to the one below to achieve this.

```
#!/bin/bash
quotacheck -vagu
```

Getting Quota Reports

The **repquota** command lists quota usage limits of all users on the system. Here is an example.

```
[root@bigboy tmp]# repquota /home
*** Report for user quotas on device /dev/hda3
Block grace time: 7days; Inode grace time: 7days
                        Block limits                File limits
User            used    soft    hard  grace    used  soft  hard  grace
----------------------------------------------------------------------
root      --   52696       0       0           1015     0     0
…
…
…
mp3user   --      24       0       0              7     0     0

[root@bigboy tmp]#
```

Chapter 5

Remote Disk Access With NFS

= =

In This Chapter

= =

Samba is usually the solution of choice when you want to share disk space between Linux and Windows machines. NFS is used when disks need to be shared between Linux servers. Directories on the NFS server are presented to the network as special NFS filesystems and the remote NFS clients use the **mount** command to gain access to them.

Installing NFS

RedHat Linux installs NFS by default and also by default it is activated when the system boots up. You can determine whether you have NFS installed using the RPM command, the main NFS package is called “nfs-utils”.

```
[root@bigboy tmp]# rpm -qa | grep nfs
redhat-config-nfs-1.0.4-5
nfs-utils-1.0.1-3.9
[root@bigboy tmp]#
```

You will also need to have the RPC portmap package installed as well. You can use the rpm command to determine whether it’s installed on your system.

```
[root@bigboy tmp]# rpm -qa | grep portmap
portmap-4.0-54
[root@bigboy tmp]#
```

If you need to do it, installing NFS and portmap is relatively straightforward. Most RedHat Linux software products are available in the RPM format. Downloading and installing RPMs isn't hard. If you need a refresher, the Linux Home Networking eBook's chapter on RPMs covers how to do this in detail. It is best to use the latest version of NFS.

- For example, the required RedHat 9.0 RPMs as of this writing were:

```
nfs-utils-1.0.1-2.9.i386.rpm
portmap-4.0-54.i386.rpm
```

- Install the packages using the **rpm** command

```
[root@bigboy tmp]# rpm -Uvh nfs-utils-1.0.1-2.9.i386.rpm
[root@bigboy tmp]# rpm -Uvh portmap-4.0-54.i386.rpm
```

How To Get NFS Started

- Use the **chkconfig** command to configure NFS and RPC portmap to start at boot. You will also have to activate NFS file locking to reduce the risk of corrupted data.

```
[root@bigboy tmp]# chkconfig --level 35 nfs on
[root@bigboy tmp]# chkconfig --level 35 nfslock on
[root@bigboy tmp]# chkconfig --level 35 portmap on
```

- Use the init scripts in the **/etc/init.d** directory to start/stop/restart NFS and RPC portmap after booting.

```
[root@bigboy tmp]# /etc/init.d/portmap start
[root@bigboy tmp]# /etc/init.d/portmap stop
[root@bigboy tmp]# /etc/init.d/portmap restart

[root@bigboy tmp]# /etc/init.d/nfs start
[root@bigboy tmp]# /etc/init.d/nfs stop
[root@bigboy tmp]# /etc/init.d/nfs restart

[root@bigboy tmp]# /etc/init.d/nfslock start
[root@bigboy tmp]# /etc/init.d/nfslock stop
[root@bigboy tmp]# /etc/init.d/nfslock restart
```

- You can test whether the NFS is running correctly with the **rpcinfo** command. You should get a listing of running RPC programs which must include; mountd, portmapper, nfs and nlockmgr.

```
[root@silent RPMS]# rpcinfo -p localhost
   program vers proto   port
    100000    2   tcp    111  portmapper
    100000    2   udp    111  portmapper
    100003    2   udp   2049  nfs
    100003    3   udp   2049  nfs
    100021    1   udp   1024  nlockmgr
    100021    3   udp   1024  nlockmgr
    100021    4   udp   1024  nlockmgr
    100005    1   udp   1042  mountd
    100005    1   tcp   2342  mountd
    100005    2   udp   1042  mountd
    100005    2   tcp   2342  mountd
    100005    3   udp   1042  mountd
    100005    3   tcp   2342  mountd
 [root@silent RPMS]#
```

The /etc/exports File

This is the main NFS configuration file and consists of two columns. The first column lists the directories you want to make available to the network. The second column has two parts. The first part lists the networks or DNS domains that can get access to the directory, the second part lists NFS options in brackets.

In the case below we have provided:

- Read only access to the /data/RedHat9 directory to all networks
- Read/write access to the /home directory from all servers on the 192.168.1.0 network
- Read/write access to the /data/test directory from servers in the my-site.com DNS domain

In all cases we have used the sync option to ensure that file data cached in memory is automatically written to the disk after the completion of any disk data copying operation.

```
#/etc/exports
/data/RedHat9           *(ro,sync)
/home                   192.168.1.0/24(rw,sync)
/data/test              *.my-site.com(rw,sync)
```

Activating Modifications The Exports File

New Exports File

When no directories have been exported to NFS, then the "**exportfs –a**" command is used

```
[root@bigboy tmp]# exportfs -a
```

Adding A Share To An Existing Exports File

When adding a share you can use the "**exportfs –r**" command to export only the new entries.

```
[root@bigboy tmp]# exportfs -r
```

Deleting, Moving Or Modifying A Share

In this case it is best to temporarily unmount the NFS directories using the "**exportfs –ua**" command followed by the "**exportfs –a**" command.

```
[root@bigboy tmp]# exportfs -ua
[root@bigboy tmp]# exportfs -a
```

NFS And DNS

The NFS client must have a matching pair of forward and reverse DNS entries on your DNS server for NFS to function correctly. In other words, a DNS lookup on the IP address of the NFS client must return a server name that will map back to the original IP address when a DNS lookup is done on that same server name.

```
[root@silent RPMS]# host 172.16.1.201
201.1.16.172.in-addr.arpa domain name pointer 172-16-1-201.simiya.com.
[root@silent RPMS]# host 172-16-1-201.simiya.com
172-16-1-201.simiya.com has address 172.16.1.201
[root@silent RPMS]#
```

Configuring The NFS Client

Ensure Portmap Is Running

Use the **rpcinfo** command to make sure the **portmap** daemon is running. Start it if it isn't. NFS clients don't need NFS to be running to mount remote directories, but need to have **portmap** running to communicate correctly with the **NFS** server.

The /etc/fstab File

You need to edit the **/etc/fstab** file if you need the NFS directory to be made permanently available to users on the NFS. In this case we're mounting the **/data/RedHat9** directory as an NFS type filesystem on the **/mnt/nfs** mount point.

```
#/etc/fstab
#Directory                     Mount
Point     Type    Options    Dump    FSCK
172.16.1.200:/data/RedHat9  /mnt/nfs        nfs     soft       0       0
```

Mounting The NFS Directory

The next thing to do is create a mount point for the remote NFS directory on the NFS client and then use the "mount –a" command to mount the directory. Notice how before mounting there were no files visible in the **/mnt/nfs** directory, this changes after the mounting is completed,

```
[root@smallfry tmp]# mkdir /mnt/nfs
[root@smallfry tmp]# ls /mnt/nfs
[root@smallfry tmp]# mount -a
[root@smallfry tmp]# ls /mnt/nfs
ISO  ISO-RedHat  kickstart  RedHat
[root@smallfry tmp]#
```

Other NFS Considerations

Security

NFS and portmap have had a number of known security deficiencies in the past and as a result, it is not recommended to use NFS over insecure networks. NFS doesn't encrypt data and it is possible for root users on NFS clients to have root access the server's filesystems.

- Exercise caution with NFS.
- Restrict its use to secure networks
- Export only the most needed data
- Consider using read only exports whenever data updates aren't necessary.
- Use the root_squash option in /etc/exports (default) to reduce the risk of the abuse of privileges by NFS client "root" users on the NFS server.

NFS Hanging

Most NFS transactions use the UDP protocol which doesn't keep track of the state of a connection. If the remote server fails, the NFS client will sometimes not be aware of the disruption in service. If this occurs, the NFS client will wait indefinitely for the return of the server. This will also force programs relying on the same client server relationship to wait indefinitely too.

It is for this reason that it's recommended to use the "soft" option in the NFS client's **/etc/fstab** file this will cause NFS to report I/O error to the calling program after a long timeout.

You can reduce the risk of NFS hanging by taking a number of precautions:

- Run NFS on a reliable network.

- Avoid having NFS servers that NFS mount each other's filesystems or directories.
- Always use the sync option whenever possible.
- Mission critical computers shouldn't rely on an NFS server to operate.
- A hung NFS connection to a directory in your search path could cause your shell to pause at that point in the search path until the NFS session is regained. NFS mounted directories shouldn't be part of your search path.

File Locking

NFS allows multiple clients to mount the same directory but NFS has a history of not handling file locking well, though more recent versions are said to have rectified the problem. Test your network based applications thoroughly before considering using NFS.

Nesting Exports

NFS doesn't allow you to export directories that are subdirectories of directories that have already been exported unless they are on different partitions.

Limiting "root" Access

NFS doesn't allow a "root" user on a NFS client to have "root" privileges on the NFS server. This can be disabled with the no_root_squash export option in the /etc/exports file.

Chapter 6

Centralized Logins Using NIS

= =

In This Chapter

= =

Network Information Services (NIS) allows you to create user accounts that can be shared across all systems on your network. The user account is created only on the NIS server. NIS clients download the necessary username and password data from the NIS server to verify each user login.

An advantage of NIS is that users only need to change their passwords on the NIS server, instead of every system on the network. This makes NIS popular in computer training labs, distributed software development projects or any other situation where groups of people have to share many different computers.

The disadvantage is that NIS doesn't encrypt the username/password information sent to the clients with each login and all users have access to the encrypted passwords stored on the NIS server. A detailed analysis of NIS security is beyond the scope of this book, but I would suggest that you restrict its use to highly secure networks or networks where access to non NIS networks is highly restricted.

Scenario

A school wants to set up a small computer lab for its students.

- The main Linux server, "**bigboy**", has a large amount of disk space and will be used as both the NIS server and NFS based file server for the Linux PCs in the lab.
- Users logging into the PCs will be assigned "home" directories on **bigboy** and not on the PCs themselves.

- Each user's "home" directory will be automatically mounted with each user login on the PCs using NFS.
- The lab instructor wants to practice with a Linux PC named "**smallfry**" before implementing NIS on all the remaining PCs.
- The suite of NIS RPMs have been installed on the server and client. As of RedHat verion 9 these were:

```
ypserv-2.6-2
yp-tools-2.7-5
ypbind-1.11-4
```

Downloading and installing RPMs isn't hard. If you need a refresher, the chapter on RPMs in the Linux Home Networking eBook covers how to do this in detail

The lab instructor has done some research and has created the following implementation plan.

- Configure **bigboy** as an NFS server which will make its **/home** directory available to the linux workstations.
- Configure **smallfry** as an NFS client that can access **bigboy**'s **/home** directory.
- Configure bigboy as an NIS server.
- Create a user account on **bigboy** called "**nisuser**" that doesn't exist on **smallfry**. Convert the account to a NIS user account.
- Configure **smallfry** as an NIS client.
- Test a remote login from **bigboy** to **smallfry** using the username and password of the account **nisuser**.

Here is how it's done.

Configuring The NFS Server

- Edit the **/etc/exports** file to allow NFS mounts of the **/home** directory with read/write access.

```
/home                    *(rw,sync)
```

- Let NFS read the **/etc/exports** file for the new entry and make **/home** available to the network with the **exportfs** command.

```
[root@bigboy tmp]# exportfs -a
[root@bigboy tmp]#
```

- Make sure the required NFS, NFS lock and port mapper daemons are both running and configured to start after the next reboot.

```
[root@bigboy tmp]# chkconfig nfslock on
[root@bigboy tmp]# chkconfig nfs on
[root@bigboy tmp]# chkconfig portmap on
[root@bigboy tmp]# /etc/init.d/nfs start
Starting NFS services:  [  OK  ]
Starting NFS quotas: [  OK  ]
Starting NFS daemon: [  OK  ]
```

```
Starting NFS mountd: [  OK  ]
[root@bigboy tmp]# /etc/init.d/nfslock start
Starting NFS statd: [  OK  ]
[root@bigboy tmp]# /etc/init.d/portmap start
Starting portmapper: [  OK  ]
[root@bigboy tmp]#
```

Configuring The NFS Client

- Keep a copy of the old **/home** directory, and create a new directory **/nfs/home** which will be a logical link to a new **/home** directory.

```
[root@smallfry tmp]# mv /home /home.save
[root@smallfry tmp]# mkdir /nfs
[root@smallfry tmp]# mkdir /nfs/home
[root@smallfry tmp]# ln -s /nfs/home/ /home
[root@smallfry tmp]# ll /
total 201
drwxr-xr-x    2 root   root   4096 Nov 15 11:19 bin
…
…
lrwxrwxrwx    1 root   root     11 Nov 16 20:22 home -> /nfs/home/
drwxr-xr-x    2 root   root   4096 Jan 24  2003 home.save
…
…
drwxr-xr-x    3 root   root   4096 Nov 16 20:19 nfs
[root@smallfry tmp]#
```

- Make sure you can mount **bigboy**'s **/home** directory on the new **/nfs/home** directory we just created. Unmount it once everything looks correct.

```
[root@smallfry tmp]# mount 172.16.1.200:/home /nfs/home/
[root@smallfry tmp]# ls /home
ftpinstall  nisuser  quotauser  smallfry  www
[root@smallfry tmp]# umount /nfs/home/
[root@smallfry tmp]#
```

- Start configuring autofs automounting. Edit your **/etc/auto/master** file to refer to file **/etc/auto.nfs** for mounting information whenever the **/nfs** directory is accessed. After five minutes, **autofs** will unmount the directory.

```
#/etc/auto.master
/nfs      /etc/auto.nfs --timeout 600
```

- Edit file **/etc/auto.nfs** to do the NFS mount whenever the /home directory is accessed.

```
#/etc/auto.nfs
home    -fstype=nfs     172.16.1.200:/home
```

- Start **autofs** and make sure it will start after the next reboot with the **chkconfig** command.

```
[root@smallfry tmp]# chkconfig autofs on
[root@smallfry tmp]# /etc/init.d/autofs restart
Stopping automount:[  OK  ]
Starting automount:[  OK  ]
[root@smallfry tmp]#
```

- Test to see whether accessing the **/home** directory will allow you to see the contents of **/home** on **bigboy**.

```
[root@smallfry tmp]# ls /home
ftpinstall  nisuser  quotauser  smallfry  www
[root@smallfry tmp]#
```

Configuring The NIS Server

In the early days, NIS was called “Yellow Pages”. The developers had to change the name after a copyright infringement lawsuit, yet many of the key programs associated with NIS have kept their original names beginning with “yp”.

Start The Key NIS Server Related Daemons

Start the necessary NIS daemons in the **/etc/init.d** directory and use the **chkconfig** command to ensure they start after the next reboot.

Daemon Name	Purpose
Portmap	The foundation RPC daemon upon which NIS runs.
Yppasswdd	Lets users change their passwords on the NIS server from NIS clients
Ypserv	Main NIS server daemon
Ypbind	Main NIS client daemon
Ypxfrd	Used to speed up the transfer of very large NIS maps

```
[root@bigboy tmp]# /etc/init.d/portmap start
[root@bigboy tmp]# /etc/init.d/yppasswdd start
[root@bigboy tmp]# /etc/init.d/ypserv start
[root@bigboy tmp]# /etc/init.d/ypbind start
[root@bigboy tmp]# /etc/init.d/ypxfrd start
[root@bigboy tmp]#
[root@bigboy tmp]# chkconfig portmap on
[root@bigboy tmp]# chkconfig yppasswdd on
[root@bigboy tmp]# chkconfig ypserv on
[root@bigboy tmp]# chkconfig ypbind on
[root@bigboy tmp]# chkconfig ypxfrd on
[root@bigboy tmp]#
```

Make Sure The Daemons Are Running

All the NIS daemons use RPC port mapping and therefore will be listed using the "**rpcinfo"** command when they are running correctly.

```
[root@bigboy tmp]# rpcinfo -p localhost
   program vers proto   port
    100000    2   tcp    111  portmapper
    100000    2   udp    111  portmapper
    100003    2   udp   2049  nfs
    100003    3   udp   2049  nfs
    100021    1   udp   1024  nlockmgr
    100021    3   udp   1024  nlockmgr
    100021    4   udp   1024  nlockmgr
    100004    2   udp    784  ypserv
    100004    1   udp    784  ypserv
    100004    2   tcp    787  ypserv
    100004    1   tcp    787  ypserv
    100009    1   udp    798  yppasswdd
 600100069    1   udp    850  fypxfrd
 600100069    1   tcp    852  fypxfrd
    100007    2   udp    924  ypbind
    100007    1   udp    924  ypbind
    100007    2   tcp    927  ypbind
    100007    1   tcp    927  ypbind
[root@bigboy tmp]#
```

Edit Your etc/sysconfig/network File

You need to add the NIS domain you wish to use in the **/etc/sysconfig/network** file. In the case below, we've called the domain "NIS-HOME_NETWORK".

```
#/etc/sysconfig/network
NISDOMAIN="NIS-HOME-NETWORK"
```

Edit Your etc/yp.conf File

NIS servers also have to be NIS clients themselves, so you'll have to edit the NIS client configuration file **/etc/yp.conf** to list the domain's NIS server as being the server itself or "localhost".

```
# /etc/yp.conf - ypbind configuration file
ypserver 127.0.0.1
```

Initialize Your NIS Domain

Now that you have decided on the name of the NIS domain, you'll have to use the **ypinit** command to create the associated authentication files for the domain. You will be prompted for the name of the NIS server, which in this case is "**bigboy**".

With this procedure, all non privileged accounts will automatically be accessible via NIS.

```
[root@bigboy tmp]# /usr/lib/yp/ypinit -m

At this point, we have to construct a list of the hosts which will run
NIS
servers.  bigboy is in the list of NIS server hosts.  Please continue to
add
the names for the other hosts, one per line.  When you are done with the
list, type a <control D>.
        next host to add:  bigboy
        next host to add:
The current list of NIS servers looks like this:

bigboy

Is this correct?  [y/n: y]  y
We need a few minutes to build the databases...
Building /var/yp/NIS-HOME-NETWORK/ypservers...
Running /var/yp/Makefile...
gmake[1]: Entering directory `/var/yp/NIS-HOME-NETWORK'
Updating passwd.byname...
Updating passwd.byuid...
Updating group.byname...
Updating group.bygid...
Updating hosts.byname...
Updating hosts.byaddr...
Updating rpc.byname...
Updating rpc.bynumber...
Updating services.byname...
Updating services.byservicename...
Updating netid.byname...
Updating protocols.bynumber...
Updating protocols.byname...
Updating mail.aliases...
gmake[1]: Leaving directory `/var/yp/NIS-HOME-NETWORK'

bigboy has been set up as a NIS master server.

Now you can run ypinit -s bigboy on all slave server.
[root@bigboy tmp]#
```

Adding New NIS Users

New NIS users can be created by logging into the NIS server and creating the new user account. In this case we'll create a user account called "**nisuser**" and give it a new password.

Once this is complete, you will then have to update the NIS domain's authentication files by executing the make command in the **/var/yp** directory.

This procedure will make all NIS enabled, non privileged accounts become automatically accessible via NIS, not just newly created ones. It will also export all the user's characteristics stored in the **/etc/passwd** and **/etc/group** files such as the login shell, the user's group and home directory.

```
[root@bigboy tmp]# useradd –g users nisuser
[root@bigboy tmp]# passwd nisuser
Changing password for user nisuser.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
```

```
[root@bigboy tmp]# cd /var/yp
[root@bigboy yp]# make
gmake[1]: Entering directory `/var/yp/NIS-HOME-NETWORK'
Updating passwd.byname...
Updating passwd.byuid...
Updating netid.byname...
gmake[1]: Leaving directory `/var/yp/NIS-HOME-NETWORK'
[root@bigboy yp]#
```

You can check to see if the user's authentication information has been updated by using the **ypmatch** command which should return the user's encrypted password sting.

```
[root@bigboy tmp]# ypmatch nisuser passwd
nisuser:$1$Cs2GMe6r$gRVB1hohkyG7ALrDLjH1:505:100::/home/nisuser:/bin/
bash
[root@bigboy tmp]#
```

Configuring The NIS Client

Run authconfig

The **authconfig** program will automatically configure your NIS files after prompting you for the IP address and domain of the NIS server.

```
[root@smallfry tmp]# authconfig
```

Once finished, it should create **/etc/yp.conf** and **/etc/sysconfig/network** files that look like this:

```
# /etc/yp.conf - ypbind configuration file
domain NIS-HOME-NETWORK server 172.16.1.200

#/etc/sysconfig/network
NISDOMAIN=NIS-HOME-NETWORK
```

Update Your /etc/nsswitch.conf File

The **/etc/nisswitch.conf** file lists the order in which certain data sources should be searched for name lookups like those in DNS and NIS. The default file tends to favor searching through local system files first. With NIS clients, you want these searches to be via NIS first, and therefore most of the entries in the file need to list NIS first.

Fortunately NIS comes with a good template file to use. You can use the "**locate**" command to find it and then you can replace the default **nsswitch.conf** file with the template as we have done below. Remember to save the old version of the file just in case.

```
[root@smallfry tmp]# locate nsswitch.conf
/etc/nsswitch.conf
/usr/share/doc/yp-tools-2.7/nsswitch.conf
[root@smallfry tmp]# cp /etc/nsswitch.conf /etc/nsswitch.conf.old
```

```
[root@smallfry tmp]# cp /usr/share/doc/yp-tools-2.7/nsswitch.conf
\
 /etc/nsswitch.conf
[root@smallfry tmp]#
```

Update Your /etc/hosts.conf File

You'll need to add "nis" to your **/etc/hosts.conf** file.

```
#/etc/host.conf
order hosts,bind,nis
```

Add Plus Entries To /etc Authentication Files

In some versions of NIS you can place what are called "plus entries" in the **/etc/passwd** file of the ***server and client*** machines to determine which user accounts will have their NIS passwords exported. All user entries below the plus entry are not exported.

This doesn't seem to apply to the RedHat version of NIS and all non privileged accounts have their passwords exported with the "**make**" command.

Non RedHat "Plus Entry" Formats

/etc/passwd

The following entry should be added to the end of the file. A "+" followed by six ":"s

```
+::::::
```

/etc/group

The following entry should be added to the end of the file. A "+" followed by three ":"s

```
+:::
```

/etc/shadow

The following entry should be added to the end of the file. A "+" followed by eight ":"s

```
+::::::::
```

Start The NIS Client Related Daemons

Start the **ypbind** NIS client, **yppasswd** and **portmap** daemons in the **/etc/init.d** directory and use the **chkconfig** command to ensure they start after the next reboot. Remember to use the "**rpcinfo**" command to ensure they are running correctly.

```
[root@smallfry tmp]# /etc/init.d/portmap start
[root@smallfry tmp]# /etc/init.d/ypbind start
[root@smallfry tmp]# /etc/init.d/yppasswdd start
```

```
[root@smallfry tmp]# chkconfig ypbind on
[root@smallfry tmp]# chkconfig portmap on
[root@smallfry tmp]# chkconfig yppasswdd on
```

Test Access to The NIS Server

Using the ypcat And getent Commands

- A good test of NIS functionality is to see whether you can get a listing of user records from the NIS server using the "**ypcat**" command.

```
[root@smallfry tmp]# ypcat passwd
nisuser:$1$Cs2GMe6r$gRZ3VB1hohkyG7ALrDLjH1:505:100::/home/nis
user:/bin/bash
quotauser:!!:503:100::/home/quotauser:/bin/bash
ftpinstall:$1$8WjAVtes$SnRh9S1w07sYkFNJwpRKa.:502:100::/:/bin
/bash
www:$1$DDCi/OPI$hwiTQ.L0XqYJUk09Bw.pJ/:504:100::/home/www:/bi
n/bash
smallfry:$1$qHni9dnR$iKDs7gfyt..BS9Lry3DAq.:501:100::/:/bin/b
ash
[root@smallfry tmp]#
```

- The getent command will also provide similar information on a "per user" basis.

```
[root@smallfry tmp]# getent passwd nisuser
nisuser:$1$Cs2GMe6r$gRZ3VB1hohkyG7ALrDLjH1:505:100::/home/nis
user:/bin/bash
[root@smallfry tmp]#
```

Logging In Via Telnet

- Try logging into the NIS client via telnet if it is enabled

```
[root@bigboy tmp]# telnet 172.16.1.201
Trying 172.16.1.201...
Connected to 172.16.1.201.
Escape character is '^]'.
Red Hat Linux release 9 (Shrike)
Kernel 2.4.20-6 on an i686
login: nisuser
Password:
Last login: Sun Nov 16 22:03:51 from 172-16-1-100.simiya.com
[nisuser@smallfry nisuser]$
```

Logging In Via SSH

- Try logging into the NIS client via SSH.

```
[root@bigboy tmp]# ssh -l nisuser  172.16.1.201
nisuser@172.16.1.201's password:
[nisuser@smallfry nisuser]$
```

- **Note:** Sometimes SSH logins on the client will fail right after activating NIS. SSH on the nIS client will need to be restarted so that it can read the new **nsswitch.conf** file and know that it should use NIS as an authentication option. Typical errors can be seen below:

Failed SSH login attempt

```
[root@bigboy tmp]# ssh -l nisuser  172.16.1.201
nisuser@172.16.1.201's password:
Permission denied, please try again.
nisuser@172.16.1.201's password:
```

Syslog Error for failed login attempt

```
Nov 16 15:29:59 smallfry sshd[3379]: Illegal user nisuser
from 172.16.1.100
```

Change Your NIS passwords

Test to make sure your users can change their NIS passwords with the **yppasswd** command.

Possible Errors

- The yppasswdd daemon must be running for this to be done or else you will get the error below.

```
[root@smallfry etc]# yppasswd -p nisuser
yppasswd: yppasswdd not running on NIS master host ("silent").
[root@smallfry etc]#
```

- There is also a known bug with this program which can cause a segmentation fault. An alternative would be to let your users change their passwords on the NIS server followed by the **root** user running the **make** command in the **/var/yp** directory as was done in the “Adding Users” section above.

```
[root@smallfry root]# yppasswd -p nisuser
Segmentation fault
[root@smallfry root]#
```

Chapter 7

Configuring Squid

= =

In This Chapter

= =

Two important goals of many small businesses are:

- Reduce their Internet bandwidth charges
- Limit access to the Web to only authorized users.

The Squid web caching proxy server can achieve both these goals fairly easily.

Users configure their web browsers to use the Squid proxy server instead of going to the web directly. The Squid server then checks its web cache for the web information requested by the user. It will return any matching information that finds in its cache, and if not will go to the web to find it on behalf of the user. Once it finds the information, it will populate its cache with it and also forward it to the user's web browser.

As you can see, this reduces the amount of data accessed from the web. Another advantage is that you can configure your firewall to only accept HTTP web traffic from the Squid server and no one else. Squid can then be configured to request usernames and passwords for each user that users its services. This provides simple access control to the Internet.

Download and Install The Squid Package

Most RedHat Linux software products are available in the RPM format. Downloading and installing RPMs isn't hard. If you need a refresher, the chapter on RPMS in the Linux Home Networking eBook covers how to do this in detail. It is best to use the latest version of Squid.

- For example, the RedHat 9.0 RPM as of this writing was:

```
squid-2.5.STABLE1-2.i386.rpm
```

- Install the package using the **rpm** command

```
[root@bigboy tmp]# rpm -Uvh squid-2.5.STABLE1-2.i386.rpm
```

The /etc/squid/squid.conf File

The main Squid configuration file is **squid.conf**. Squid must be restarted each time you make changes to this file for them to come into effect.

The Visible Host Name

Note: Squid will fail to start if you don't give your server a hostname. You can set this with the "visible_hostname" parameter. Here we set it to the real name of our server "bigboy".

```
visible_hostname bigboy
```

Restricting Web Access By Time

Access control lists can be created with time parameters. Here are some quick examples. Remember to restart Squid for the changes to take effect.

Only Allow Business Hour Access From The Home Network

```
#
# Add this to the bottom of the ACL section of squid.conf
#
acl home_network src 192.168.1.0/24
acl business_hours time M T W H F 9:00-17:00
#
# Add this at the top of the http_access section of squid.conf
#
http_access allow home_network business_hours
```

Only Allow Access In The Morning

```
#
# Add this to the bottom of the ACL section of squid.conf
#
acl mornings time 08:00-12:00

#
# Add this at the top of the http_access section of squid.conf
#
http_access allow mornings
```

Restricting Web Access By IP Address

You can create an access control list (ACL) that restricts web access to users on certain networks. In this case we're creating an ACL that defines our home network of 192.168.1.0.

```
#
# Add this to the bottom of the ACL section of squid.conf
#
acl home_network src 192.168.1.0/255.255.255.0
```

You will also have to add a corresponding **http_access** statement that allows traffic that matches the ACL.

```
#
# Add this at the top of the http_access section of squid.conf
#
http_access allow home_network
```

Remember to restart Squid for the changes to take effect.

Password Authentication Using NCSA

Squid can be configured to prompt users for a username and password. Squid comes with a program called "**ncsa_auth**" that will read any NCSA compliant encrypted password file. The "**htpasswd"** program that comes installed with Apache can be used to create your passwords. Here is how it's done:

Create The Password File

```
[root@bigboy tmp]# touch /etc/squid/squid_passwd
[root@bigboy tmp]# chmod o+r /etc/squid/squid_passwd
```

Add Users To The Password File

Use the htpasswd program to add users to the password file. In this case we add a username called "www". You can add users at anytime without having to restart Squid.

```
[root@bigboy tmp]# htpasswd /etc/squid/squid_passwd www
New password:
Re-type new password:
Adding password for user www
[root@bigboy tmp]#
```

Locate Your ncsa_auth File

- First you need to locate were **ncsa_auth** is located with the locate command.

```
[root@silent RPMS]# locate ncsa_auth
/usr/lib/squid/ncsa_auth
[root@silent RPMS]#
```

Edit squid.conf

- You need to define the authentication program, which is in this case **ncsa_auth**.
- The next step is to create an "**http_access**" entry that allows traffic that matches a special access control list (ACL) entry we'll call "**ncsa_users**".
- The next step create the ACL named "**ncsa_users**" with the REQUIRED keyword that forces Squid to use the NCSA **auth_param** method we defined.

Simple User Authentication Example

```
#
# Add this to the auth_param section of squid.conf
#
auth_param basic program /usr/lib/squid/ncsa_auth
/usr/etc/passwd

#
# Add this to the bottom of the ACL section of squid.conf
#
acl ncsa_users proxy_auth REQUIRED
#
# Add this at the top of the http_access section of
squid.conf
#
http_access allow ncsa_users
```

Password Authentication, Access only during Business hours

```
#
# Add this to the auth_param section of squid.conf
#
auth_param basic program /usr/lib/squid/ncsa_auth
/usr/etc/passwd

#
# Add this to the bottom of the ACL section of squid.conf
#
acl ncsa_users proxy_auth REQUIRED
acl business_hours time M T W H F 9:00-17:00


#
# Add this at the top of the http_access section of
squid.conf
#
http_access allow ncsa_users business_hours
```

Restart Squid

Remember to restart Squid for the changes to take effect.

Configuring Web Browsers To Use Your Squid Server

You can configure your browser's proxy server using the following methods:

Internet Explorer

Click on the "Tools" item on the menu bar of the browser.

- Click on "Internet Options"
- Click on "Connections"
- Click on "LAN Settings"
- Configure with the address and TCP port (3128 default) used by your Squid server.

Mozilla / Netscape

Click on the "Edit" item on the menu bar of the browser.

- Click on "Preferences"
- Click on "Advanced"
- Click on "Proxies"
- Configure with the address and TCP port (3128 default) used by your Squid server under "Manual Proxy Configuration"

How To Get Squid Started

- Use the chkconfig configure Squid to start at boot:

```
[root@bigboy tmp]# chkconfig --level 35 squid on
```

- Use the squid init script in the **/etc/init.d** directory to start/stop/restart Squid after booting

```
[root@bigboy tmp]# /etc/init.d/squid start
[root@bigboy tmp]# /etc/init.d/squid stop
[root@bigboy tmp]# /etc/init.d/squid restart
```

- You can test whether the Squid process is running with the following command, you should get a response of plain old process ID numbers:

```
[root@bigboy tmp]# pgrep squid
```

Squid And Firewalls

If you are using access controls on Squid, you may want to restrict Internet access to only the Squid server. Also, if there is a firewall between your Squid server and the client PCs you will need to allow TCP port 3128 traffic through.

Squid Disk Usage

Squid uses the **/var/spool/squid** directory to store its cache files. High usage squid servers will most likely need a large amount of disk space in the **/var** partition to get optimum performance.

Every webpage and image accessed via the Squid server is logged in the **/var/log/squid/access.log** file. This can get quite large on high usage servers. Fortunately, the **logrotate** program automatically purges this file.

Troubleshooting Squid

Squid logs both informational and error messages to files in the **/var/log/squid/** directory. It is best to review these files first whenever you have difficulties.

Other Squid Capabilities

The following capabilities are beyond the scope of this eBook, but have been included to provide a broader understanding.

- For performance reasons, you can configure “child” Squid servers on which certain types of content are exclusively cached.
- You can restrict the amount of disk space and bandwidth used by Squid.
- You can set up your network such that users don’t have to modify their browser settings. This is called a “transparent proxy” configuration. It is usually achieved by configuring a firewall between the client PCs and the Squid server to redirect all port 80 traffic to the Squid server on port 3128.

Chapter 8

Modifying The Kernel To Improve Performance

= =

In This Chapter

= =

Like a government that rules a nation and all its provinces, the Linux kernel is the central program that not only governs how programs should interact with one another, but also provides the guidelines on how they should use the computer's core infrastructure such as memory, disks and other input/output (I/O) devices for the benefit of the user.

Linux drivers, the programs that manage each I/O device, could be viewed as being the staff that keeps all the government departments running. Continuing with the analogy, the more departments you make the kernel manage, the slower Linux becomes. Large kernels also reduce the amount of memory left over for user applications. These may then be forced to juggle their memory needs between RAM and the much slower swap partitions of disk drives causing the whole system to become sluggish.

The RedHat installation CDs have a variety of kernel RPMs, and the installation process autodetects the one most suited for your needs. It is for this reason that the RedHat Linux kernel installed on your system is probably sufficient.

However, there are times when you may want to rebuild it. For example, there is no installation RPM for multiprocessor systems with large amounts of memory. You may also want to experiment in making a high speed Linux router without support for SCSI, USB, Bluetooth and sound but with support for a few NIC drivers, an IDE hard drive, and a basic VGA console. This would require a kernel rebuild.

Table 8.1: Kernels Found On Redhat Installation CDs

Processor Type	Configuration
i386	Multiprocessor (SMP)
i586	Single processor, Large memory
i686	Single processor
Athlon	

This chapter provides an overview of how to rebuild your kernel. Always practice on a test system, and keep a backup copy of your old kernel when you decide to update a production system.

Download and Install The Kernel Sources Package

Most RedHat Linux software products are available in the RPM format. Downloading and installing RPMs isn't hard. If you need a refresher, the chapter on RPMS in the Linux Home Networking eBook covers how to do this in detail. It is best to use the latest version of Squid.

- For example, the RedHat 9.0 RPM as of this writing was:

```
kernel-source-2.4.20-6.i386.rpm
```

- Install the package using the **rpm** command

```
[root@bigboy tmp]# rpm -Uvh kernel-source-2.4.20-6.i386.rpm
```

Creating A Custom Kernel

The installation of the kernel sources creates a file called README in the **/usr/src/linux-2.4** directory which briefly outlines the steps needed to create a new kernel. A more detailed explanation of the required steps is described below.

Make Sure Your Source Files Are In Order

Cleaning up the various source files is the first step that needs to be done. This isn't so important for a first time rebuild, but is vital for subsequent attempts. The "**make mrproper**" is used to do this and must be executed in the /usr/src/linux-2.4 directory.

```
[root@smallfry linux-2.4]# cd /tmp
[root@smallfry tmp]# cd /usr/src/linux-2.4
[root@smallfry linux-2.4]# make mrproper
...
...
...
[root@smallfry linux-2.4]#
```

The ".config" File

You'll then need to run scripts to create a kernel configuration file called **/usr/src/linux-2.4/.config** which lists all the kernel options you wish to use.

Backup Your Configuration

Your First Time Creating A Custom Kernel

The **.config** file won't exist if you've never created a custom kernel on your system before but fortunately, RedHat stores a number of default **.config** files in the **/usr/src/linux-2.4/configs** directory. You can be automatically copy the .config file that matches your installed kernel by running the "**make oldconfig**" command in the **/usr/src/linux-2.4** directory as seen below.

```
[root@smallfry tmp]# cd /usr/src/linux-2.4
[root@smallfry linux-2.4]# ls .config
ls: .config: No such file or directory
[root@smallfry linux-2.4]# make oldconfig
...
...
...
[root@smallfry linux-2.4]#
```

You've Created A Custom Kernel Before

The **.config** file used by the previous custom kernel build will already exist. Copy it to a safe location before proceeding.

Customizing The ".config" File

There are a number of commands listed in table 8.3 that you can run in the **/usr/src/linux-2.4** directory to update the **.config** file.

Table 8.2: Scripts For Modifying The .config File

Command	Description
make config	Text based utility that prompts you line by line. This method can become laborious.
make menuconfig	Text menu based utility.
make xconfig	X-Windows based utility.

Each command will prompt you in different ways for each kernel option, each of which will generally provide you with the three choices shown in table 8.3. A brief description of each kernel option is given in table 8.4.

Table 8.3: Kernel Option Choices

Kernel Option Choice	Description
M	The kernel will load the drivers for this option on an as needed basis. Only the code required to load the driver on demand will be included in the kernel.
Y	Include all the code for the drivers needed for this option into the kernel itself. This will generally make the kernel larger and slower but will make it more self sufficient. The "Y" option is frequently used in cases in which a stripped down kernel is one of the only programs Linux will run, such as purpose built home firewall appliances you can buy in a store. There is a limit to the overall size of a kernel. It will fail to compile if you select parameters that will make it too big.
N	Don't make the kernel support this option at all.

Table 8.4: Kernel Configuration Options

Option	Description
Code maturity level options	Determines whether Linux will prompt you for certain types of development code or drivers.
Loadable module support	Support for loadable modules versus a monolithic kernel. Most of the remaining kernel options use loadable modules by default. It is best to leave this alone in most cases.
Processor type and features	SMP, Large memory, BIOS and CPU type settings.
General setup	Support for power management, networking, systems buses such as PCI, PCMCIA, EISA, ISA
Memory technology devices	
Parallel port support	Self explanatory
Plug and Play configuration	Self explanatory
Block devices	Support for a number of parallel port based and ATAPI type devices. Support for your loopback interface and RAM disks can be found here too.
Multi-device support (RAID, LVM)	Support for RAID, 0, 1 and 5 as well as LVM.
Cryptography support	

Option	Description
(CryptoAPI)	
Networking options	TCP/IP, DECnet, Appletalk, IPX, ATM/LANE
Telephony support	Support for voice to data I/O cards
ATA/IDE/MFM/RLL support	Support for a variety of disk controller chipsets
SCSI support	Support for a variety of disk controller chipsets. Also sets limits on the maximum number of supported SCSI disks and CDROMs.
Fusion MPT support	High speed SCSI chipset support.
I2O device support	Support for specialized Intelligent I/O cards
Network device support	Support for Ethernet, Fibre Channel, FDDI, SLIP, PPP, ARCnet, Token Ring, ATM, PCMCIA networking, specialized WAN cards.
Amateur Radio support	Support for packet radio
IrDA subsystem support	Infrared wireless network support
ISDN subsystem	Support for ISDN
Old CD-ROM drivers (not SCSI, not IDE)	Support for non SCSI, non IDE, non ATAPI CDROMs
Input core support	Keyboard, mouse, joystick support in addition to the default VGA resolution.
Character devices	Support for virtual terminals and various serial cards for modems, joysticks and basic parallel port printing.
Multimedia devices	Streaming video and radio I/O card support
Crypto Hardware support	Web based SSL hardware accelerator card support
Console drivers	Support for various console video cards
Filesystems	Support for all the various filesystems and strangely, the native languages supported by Linux.
Sound	Support for a variety of sound cards
USB support	Support for a variety of USB devices
Additional device driver support	Miscellaneous driver support
Bluetooth support	Support for a variety of Bluetooth devices
Kernel hacking	Support for detailed error messages for persons writing device drivers

Configure Dependencies

As stated before, the **.config** file you just created lists the options you'll need in your kernel. The next step is prepare the needed source files for compiling with the "**make dep**" command.

```
[root@smallfry linux-2.4]# make dep
…
…
…
[root@smallfry linux-2.4]#
```

Edit The Makefile To Give The Kernel A Unique Name

Edit the file Makefile and change the line "EXTRAVERSION =" to create a unique suffix at the end of the default name of the kernel.

For example, if your current kernel version is 2.4.20-6, and your EXTRAVERSION is set to "-6-new", your new additional kernel will have the name "**vmlinuz-2.4.20-6-new**"

Remember to change this for each new version of the kernel you create.

Compile A New Kernel

The "**make bzImage**" command can now be used to create a compressed version of your new kernel. This could take several hours on a 386 or 486 system. It will take about 20 minutes on a 400MHz Celeron.

```
[root@smallfry linux-2.4]# make bzImage
…
…
…
[root@smallfry linux-2.4]#
```

Build The Kernel's Modules

The "**make modules**" command can now be used to create all the modules the kernel will need.

```
[root@smallfry linux-2.4]# make modules
…
…
…
[root@smallfry linux-2.4]#
```

Install the Kernel Modules

You'll now need to install the kernel modules you previously created once the new kernel is in place.

```
[root@smallfry linux-2.4]# make modules_install
…
…
…
[root@smallfry linux-2.4]#
```

Copy The New Kernel To The /boot Partition

The kernel you just created needs to be copied to the **/boot** partition where all your systems active kernel files normally reside. This is done with the "**make install**" command.

This partition has a default size of 100 MB which is enough to hold a number of kernels. You may have to delete some older kernels in order to create enough space.

```
[root@smallfry linux-2.4]# make install…
…
…
[root@smallfry linux-2.4]#
```

Here you can see that the new kernel "vmlinuz-2.4.20-6-new" has been installed in the **/boot** directory.

```
[root@smallfry linux-2.4]# ls -l /boot/vmlinuz*
lrwxrwxrwx 1 root root       22 Nov 28 01:20 /boot/vmlinuz -> vmlinuz-
2.4.20-6-new
-rw-r--r-- 1 root root  1122363 Feb 27  2003 /boot/vmlinuz-2.4.20-6
-rw-r--r-- 1 root root  1122291 Nov 28 01:20 /boot/vmlinuz-2.4.20-6-new
[root@smallfry linux-2.4]#
```

Updating GRUB

You should now update your **/etc/grub.conf** file to include an option to boot the new kernel. The "make install" command will do this for you automatically.

In this example, "default" is set to "1", which means the system will boot the second kernel entry, which happens to be that of the original kernel 2.4.20-6. You can set this value to "0" which will make it boot your newly compiled kernel which is the first entry.

```
default=1
timeout=10
splashimage=(hd0,0)/grub/splash.xpm.gz
title Red Hat Linux (2.4.20-6-new)
        root (hd0,0)
        kernel /vmlinuz-2.4.20-6-new ro root=LABEL=/
        initrd /initrd-2.4.20-6-new.img
title Red Hat Linux (2.4.20-6)
        root (hd0,0)
        kernel /vmlinuz-2.4.20-6 ro root=LABEL=/
        initrd /initrd-2.4.20-6.img
```

Creating A Boot Diskette For The New Kernel

You can create a rescue diskette with your new kernel with the **mkbootdisk** command.

```
[root@smallfry tmp]# mkbootdisk 2.4.20-6-new
```

Updating The Kernel Using RPMs

It is also possible to install a new standardized kernel from an RPM file. As you can see, it is much It is much simpler than crea

Creating an additional kernel using RPMs

```
[root@smallfry tmp]# rpm –ivh kernel-file.rpm
```

Replacing an existing kernel using RPMs

```
[root@smallfry tmp]# rpm –Uvh kernel-file.rpm
```

Chapter 9

Configuring Linux VPNs

==========

In This Chapter

==========

As your SOHO grows, you'll eventually need to establish some form of VPN link with a supplier, vendor, business partner or customer so that you can directly and freely access all their servers behind their firewall.

A VPN can be really convenient as you'll be able refer to the remote servers, not by their public NAT-ted IP addresses, but by their real private IP addresses. This avoids problems inherent in connecting to servers behind a "many to one" NAT configuration.

This chapter will outline the configuration of a permanent site to site VPN link or "tunnel" using FreeS/WAN, one of the most popular VPN packages for Linux.

VPN Guidelines

There are some recommended guidelines I'd suggest before attempting a simple SOHO Linux VPN.

- Make sure that the VPN traffic will not pass through a firewall that does NAT. Having a firewall as the VPN "start" or "end" point is OK, just don't make the tunnel pass through NAT as NAT breaks VPNs.
- Life will be much easier if you make your Linux VPN box also function as a firewall. Configure and test the firewall first and then configure the VPN. The chapter on the iptables firewall should help a lot.
- Take a quick refresher on the main VPN terms.

Scenario

In this example we have two SOHO offices. For simplicity, both sites use IP addressing schemes on their protected networks that do not overlap.

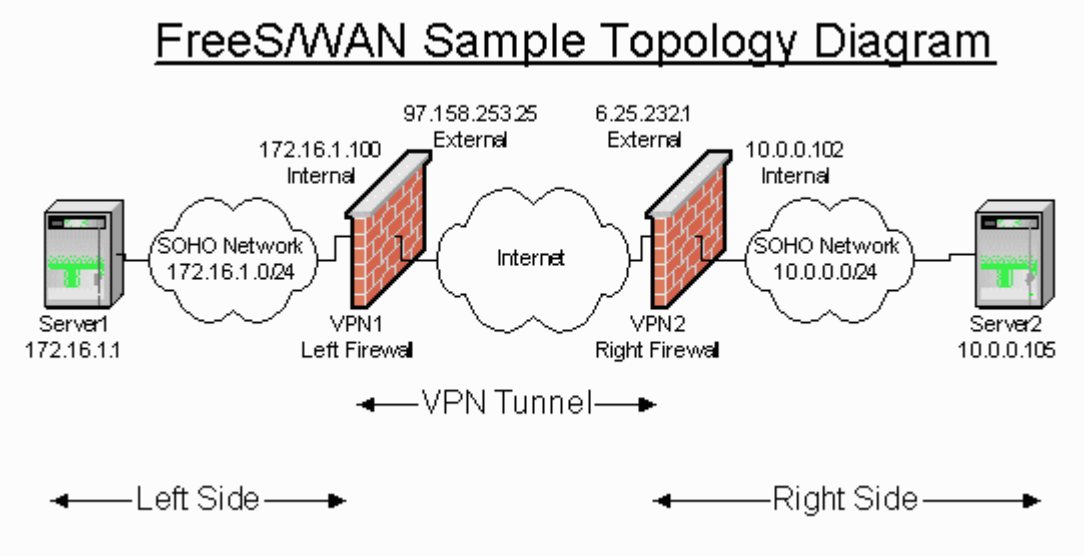
- A VPN needs to be created between the two sites so that they can communicate with each other without the fear of eavesdropping.
- For simplicity, neither site is site wants to invest in a CA certificate service or infrastructure. The RSA key encryption methodology will be used for key exchange. An alternative Cisco compatible “shared secret” (also known as a “pre-shared” or even a “symmetric” key) method will also be discussed at the end of the chapter.
- The network administrators at both sites are aware that permanent site – to – site VPNs require fixed Internet IP addresses and have upgraded from their basic DHCP services originally provided by their ISPs.

Site 1

- uses a private network of 172.168.1.0
- has a Linux VPN / firewall device default gateway with an external Internet IP address of 97.158.253.25

Site 2

- uses a private network of 10.0.0.0
- has a Linux VPN / firewall device default gateway with an external Internet IP address of 6.25.232.1



Download And Install The FreeS/WAN Package

The FreeS/WAN RPM can be downloaded from the website http://www.freeswan.org which has good instructions on how to install the product on RedHat and other versions of Linux. One of the requirements for downloading the RedHat RPM version of FreeS/WAN is to have the **ncftp** RPM package installed on your system.

Note: Most RedHat Linux software products are available in the RPM format. Downloading and installing RPMs isn't hard. If you need a refresher, the chapter on RPMs covers how to do this in detail.

Installing the Prerequisite ncftp RPM Package

Use the **rpm -qa** command to see whether you have the package installed. You'll need to install it if you get no response from the command below.

ncftp Installed

```
[root@vpn1 tmp]# rpm -qa | grep ncftp
ncftp-3.1.3-6
[root@vpn1 tmp]#
```

ncftp not Installed

```
[root@vpn1 tmp]# rpm -qa | grep ncftp
[root@vpn1 tmp]#
```

- The latest version of the RPM for RedHat 9.0 is:

```
ncftp-3.1.5-4.i386.rpm
```

- Install the package using the following command:

```
[root@bigboy tmp]# rpm -Uvh ncftp-3.1.5-4.i386.rpm
```

Installing The FreeS/WAN RPM

Once you have installed **ncftp**, you can install FreeS/WAN using the recommended commands on the website.

Downloading

Make sure you are in the directory you normally use to download RPMs and then execute the command below. (The command is actually all on one line)

```
[root@vpn2 tmp]# ncftpget
ftp://ftp.xs4all.nl/pub/crypto/freeswan/binaries/RedHat-RPMs/`uname -r |
tr -d 'a-wy-z'`/\*

freeswan-allkeys:                              132.14 kB   88.72 kB/s
freeswan-allkeys.sig:                          460.00 B     2.46 kB/s
```

```
freeswan-module-2.00_2.4.18_14-0.i386.rpm:     871.71 kB  138.08 kB/s
freeswan-rpmsign.asc:                            1.05 kB    5.78 kB/s
freeswan-sigkey.asc:                             1.62 kB    8.75 kB/s
freeswan-userland-2.00_2.4.18_14-0.i386.rpm:     1.18 MB  143.17 kB/s
[root@vpn2 tmp]#
```

Installing

Use the **rpm –ivh** command to install the two RPM packages you just downloaded.

```
[root@vpn2 tmp]# rpm -ivh freeswan*.rpm
warning: freeswan-module-2.00_2.4.18_14-0.i386.rpm: V3 RSA/MD5 signature:
NOKEY, key ID 5a7e4731
Preparing...         ########################################### [100%]
   1:freeswan-module ########################################### [ 50%]
do not forget to install the userland utilities
   2:freeswan-userland  ########################################### [100%]
invoke "service ipsec start" or reboot to begin
[root@vpn2 tmp]#
```

Starting FreeS/WAN For The First Time

The command below will start FreeS/WAN, but remember you can use the scripts in the **/etc/init.d** directory to do the same.

```
[root@vpn2 tmp]# service ipsec start
ipsec_setup: Starting FreeS/WAN IPsec 2.00...
ipsec_setup: insmod: ipsec: no module by that name found
ipsec_setup: insmod failed, but found matching template module 30ae280d.
ipsec_setup: Copying /lib/modules/2.4.18-14/kernel/net/ipsec/30ae280d to
/lib/modules/2.4.18-14/kernel/net/ipsec/ipsec.o.
ipsec_setup: /sbin/insmod /lib/modules/2.4.18-
14/kernel/net/ipsec/ipsec.o
ipsec_setup: Using /lib/modules/2.4.18-14/kernel/net/ipsec/ipsec.o
ipsec_setup: Symbol version prefix ''
ipsec_setup: WARNING: changing route filtering on wlan0 (changing
/proc/sys/net/ipv4/conf/wlan0/rp_filter from 1 to 0)

[root@vpn2 tmp]#
```

Get The Status Of The FreeS/WAN Installation

The **ipsec verify** command should give an [OK] status for most of its checks like the one below:

```
[root@vpn2 tmp]# ipsec verify
Checking your system to see if IPsec got installed and started
correctly
Version check and ipsec on-path                                 [OK]
Checking for KLIPS support in kernel                            [OK]
Checking for RSA private key (/etc/ipsec.secrets)               [OK]
Checking that pluto is running                                  [OK]
DNS checks.
Looking for forward key for vpn2                              [NO KEY]
Does the machine have at least one non-private address          [OK]
Two or more interfaces found, checking IP forwarding            [OK]
Checking NAT and MASQUERADING                                   [OK]
[root@vpn2 tmp]#
```

FreeS/WAN Configuration Steps

FreeS/WAN is very forgiving when it establishes a tunnel. It will automatically go through all the various combinations of IKE & IPSec settings with the remote VPN box until it finds a match. You don't have to configure most of these settings explicitly as you often have to do in the case of routers & firewall/VPN appliances.

Preparation work requires you to draw a basic network diagram like the one above. The VPN box on the left will be called "the left hand side" and the one on the right will be called "the right hand side".

Parameter	Description
Left	• Internet IP address of the left hand side VPN device
Leftsubnet	• The network protected by the left hand side VPN device
Leftid	• Fully Qualified Domain Name in DNS of the left hand side VPN device preceded by an "@" sign. They can be fictitious, as long as the "left" and "right" ones are unique.
Leftrsasigkey	• The entire "left" RSA sig public key for the left hand side VPN device. This can be obtained by using the "**ipsec showhostkey – left**" command.
Leftnexthop	• The next hop router from the left hand side VPN device when trying to reach the right hand side VPN device. You may use an auto-generated variable "%defaultroute" which will be valid in most cases, or the actual IP address of the next hop router in cases where the next hop is not the default router.
Right	• Internet IP address of the right hand side VPN device
Rightsubnet	• The network protected by the right hand side VPN device
Rightid	• Fully Qualified Domain Name in DNS of the right hand side VPN device preceded by an "@" sign. They can be fictitious, as long as the "left" and "right" ones are unique.
Rightrsasigkey	• The entire "right" RSA sig public key for the right hand side VPN device. This can be obtained by using the "**ipsec showhostkey – right**" command.
Rightnexthop	• The next hop router from the right hand side VPN device when trying to reach the right hand side VPN device. You may use an auto-generated variable "%defaultroute" which will be valid in most cases, or the actual IP address of the next hop router in cases where the next hop is not the default router.

Once you have all this information you'll have to enter it in the **/etc/ipsec.conf** configuration file. The steps to do this will follow.

Get The RSA Keys

In order to configure the **/etc/ipsec.conf** file you'll need to get the "left" RSA public key for the "left" VPN device and the "right" key for the "right" VPN device. You'll need to take note of these and insert them in the **/etc/ipsec.conf** file.

The FreeS/WAN installation automatically generates the keys, but if you want to change them you can do so by issuing the following command:

```
[root@vpn2 tmp]# ipsec rsasigkey --verbose 2048 > keys.tmp
getting 128 random bytes from /dev/random...
looking for a prime starting there (can take a while)...
found it after 129 tries.
getting 128 random bytes from /dev/random...
looking for a prime starting there (can take a while)...
found it after 662 tries.
computing modulus...
computing lcm(p-1, q-1)...
computing d...
computing exp1, exp1, coeff...
output...
[root@vpn2 tmp]#
```

You can then edit the **/etc/ipsec.secrets** file and replace the contents between the "**RSA: {**" and the final "**}**" with the contents of the keys.tmp file generated from the **ipsec** command above.

Note: If you cut and paste these keys from the screen into Microsoft "notepad" you'll find that it may automatically insert carriage return and line feed characters at the end of each line where the text would normally wrap around on the screen. This will corrupt the keys. My experience has been much better cutting and pasting into Microsoft Word or Linux "vi" which behave better. To be safe, just make sure that the all the characters are there especially those at the far left and far right of the "cut" screen and "paste" screen.

Get The Left Key

```
[root@vpn1 tmp]# ipsec showhostkey --left
        # RSA 2192 bits   vpn1   Mon Jun 16 21:15:31 2003
        leftrsasigkey=0sAQNrV9AYdaW94FXvIxu5p54+MRaW0wy0+HHQrdGo
fklZYQ4TCBlL+Ym00Ahfc8mqXlerZY12Os41G8SIV+zzIO04WZ4wmOvEr8DZaldT
bfCuvUvMhrTtCpZdm53yF5rCaUbg+Vmx71jcIVZqwd2AAocrthuC1riwI8yK9HrS
VHzpNQxfRX+F9B//gwWEu1UVcEPkAuY2+Q2tBjg/wmFLEhOrdey7X3NRKyEa6Lqg
M2Igjhx6vflQg1ImFFyUAL37ie4YSpDnUVzy3tzBVgyPuFHOGyqZtuD/+YkNIhrH
fgyVmGu8/kuhzB7nWtOYqDFO8OHDGePOyOVPQi73KfRoDbdb3ND0EtfnRhRPblKJ
239OlIq1
[root@vpn1 tmp]#
```

Get The Right Key

```
[root@vpn2 tmp]# ipsec showhostkey --right
        # RSA 2192 bits   vpn2   Mon Jun 16 21:06:20 2003
```

```
        rightrsasigkey=0sAQNNdxFPWCga+E/AnDgIM+uIDq4UXcZzpomwMFUpyQ9+rhU
HT9w8nr3rjUR/qTZOKR2Vqd4XoBd1HkPDBQ8oNjtA3Oz+UQOU3KTMHN5ydFwe6MpTJV/hL6L
vhB0OXQad/NhjMIx8vVotlIyZJQVZlHwUevO9/C7W6+8YCfJHJitPOF2aXDT2EKdch/Dn/4p
4aiuzy/g8iwjJ+DDBxBYya9aC4GvPqSJhIiX1BwxMOOV+y5FVGs5j9wvOVpF4PJC5tayFuSa
gDFuuuNELQlQSm9gRRr/ji47xDXsmrzXOnhM8g8SPRnj7pL3abgu7Sg7eFREV1MJSVBhp0DJ
0EbVMVV+Xvwlm9++9zbY3mlc+cSXMPAJZ
[root@vpn2 tmp]#
```

Edit The /etc/ipsec.conf Configuration File

Here we have created a sub-section called “net-to-net” in which we have inserted all the needed parameters. You need to add a new sub-section for each new VPN tunnel.

```
conn net-to-net
    left=97.158.253.25           # Public Internet IP address of the
                                 # LEFT VPN device
    leftsubnet=172.16.1.0/24     # Subnet protected by the LEFT VPN device
    leftid=@vpn1.site1.com       # FQDN of Public Internet IP address of the
                                 # LEFT VPN device with an "@"

    leftrsasigkey=0sAQNrV9AYdaW94FXvIxu5p54+MRaW0wy0+HHQrdGofklZYQ4TCBlL+Ym00Ah
fc8mqXlerZY12Os41G8SIV+zzIO04WZ4wmOvEr8DZaldTbfCuvUvMhrTtCpZdm53yF5rCaUbg+Vmx71
fgyVmGu8/kuhzB7nWtOYqDFO8OHDGePOyOVPQi73KfRoDbdb3ND0EtfnRhRPblKJ239OlIq1

    leftnexthop=%defaultroute    # correct in many situations
    right=6.25.232.1             # Public Internet IP address of
                                 # the RIGHT VPN device
    rightsubnet=10.0.0.0/24      # Subnet protected by the RIGHT VPN device
    rightid=@vpn2.site1.com      # FQDN of Public Internet IP address of the
                                 # RIGHT VPN device with an "@"

    rightrsasigkey=0sAQNNdxFPWCga+E/AnDgIM+uIDq4UXcZzpomwMFUpyQ9+rhUHT9w8nr3rjU
R/qTZOKR2Vqd4XoBd1HkPDBQ8oNjtA3Oz+UQOU3KTMHN5ydFwe6MpTJV/hL6LvhB0OXQad/NhjMIx8v
OnhM8g8SPRnj7pL3abgu7Sg7eFREV1MJSVBhp0DJ0EbVMVV+Xvwlm9++9zbY3mlc+cSXMPAJZ

    rightnexthop=97.158.253.25   # correct in many situations
    auto=start                   # authorizes and starts this connection
                                  # on booting
```

Some Important Notes About The /etc/ipsec.conf File

- **Note:** It is important to maintain the indentation before each parameter as shown below.

  Right

  ```
  conn net-to-net
      left=x.x.x.x
      leftsubnet=y.y.y.y/24
  ```

  Wrong

  ```
  conn net-to-net
  left=x.x.x.x
  leftsubnet=y.y.y.y/24
  ```

- **Note:** The “net-to-net” sub sections must be the ***same*** in the **/etc/ipsec.conf** for both the left and right hand side VPN devices.

Restart FreeS/WAN

This will need to be done on both VPN devices in order for the new **/etc/ipsec.conf** settings to take effect.

```
[root@vpn2 tmp]# /etc/init.d/ipsec restart
ipsec_setup: Stopping FreeS/WAN IPsec...
ipsec_setup: Starting FreeS/WAN IPsec 2.00...
ipsec_setup: Using /lib/modules/2.4.18-14/kernel/net/ipsec/ipsec.o
[root@vpn2 tmp]#
```

Initialize The New Tunnel

You can use the **ipsec** command to start the tunnel "net-to-net" we created in the **/etc/ipsec.conf** file. You'll have to issue the command simultaneously on the VPN boxes at both ends of the tunnel or else you may get a timeout error as seen below.

```
[root@vpn2 tmp]#  ipsec auto --up net-to-net
104 "net-to-net" #1: STATE_MAIN_I1: initiate
106 "net-to-net" #1: STATE_MAIN_I2: sent MI2, expecting MR2
108 "net-to-net" #1: STATE_MAIN_I3: sent MI3, expecting MR3
003 "net-to-net" #1: discarding duplicate packet; already
STATE_MAIN_I3
010 "net-to-net" #1: STATE_MAIN_I3: retransmission; will wait 20s
for response
010 "net-to-net" #1: STATE_MAIN_I3: retransmission; will wait 40s
for response
031 "net-to-net" #1: max number of retransmissions (2) reached
STATE_MAIN_I3.  Possible authentication failure: no acceptable
response to our first encrypted message
000 "net-to-net" #1: starting keying attempt 2 of an unlimited
number, but releasing whack
[root@vpn2 tmp]#
```

Retrying the command should get it to work. The "IPsec SA established" message signifies success.

```
[root@vpn2 tmp]#  ipsec auto --up net-to-net
112 "net-to-net" #4: STATE_QUICK_I1: initiate
004 "net-to-net" #4: STATE_QUICK_I2: sent QI2, IPsec SA
established
[root@vpn2 tmp]#
```

Testing Your FreeS/WAN VPN

Check The Routes

You'll need to check the routes once the VPN tunnel is up. As you can see there is a route to the 172.16.1.0 network via the new "virtual" ipsec0 interface pointing to the VPN device at the other end of the tunnel.

```
[root@vpn2 tmp]# netstat -nr
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
10.0.0.0        0.0.0.0         255.255.255.0   U        40 0          0 eth0
6.25.232.0      0.0.0.0         255.255.255.248 U        40 0          0 wlan0
6.25.232.0      0.0.0.0         255.255.255.248 U        40 0          0 ipsec0
172.16.1.0      97.158.253.25   255.255.255.0   UG       40 0          0 ipsec0
127.0.0.0       0.0.0.0         255.0.0.0       U        40 0          0 lo
0.0.0.0         6.25.232.6      128.0.0.0       UG       40 0          0 ipsec0
128.0.0.0       6.25.232.6      128.0.0.0       UG       40 0          0 ipsec0
0.0.0.0         6.25.232.6      0.0.0.0         UG       40 0          0 wlan0
[root@vpn2 tmp]#
```

The “ipsec look” Command

You can also issue the **ipsec look** command to see whether the tunnel is up and running. You’ll have to double check your keys in **/etc/ipsec.secrets** and all the values in your **/etc/ipsec.conf** file if any of these three sections doesn’t appear.

- A section describing the existence of a phase 1 IKE connection that looks like this:

```
0.0.0.0/0           -> 0.0.0.0/0           => %trap (0)
10.0.0.0/24         -> 172.16.1.0/24       =>
tun0x1004@97.158.253.25 esp0x1d096702@97.158.253.25  (4)
6.25.232.1/32       -> 0.0.0.0/0           => %trap (4)
```

  Here we can see some form of connectivity between the two networks at either end of the tunnel namely 10.0.0.0/24 and 172.16.1.0/24.

- A section that describes the state of the phase 2 IPSEC tunnel. There will be many “esp” references involving the IP addresses of the VPN devices at either end of the tunnel.

```
esp0x1d096701@97.158.253.25 ESP_3DES_HMAC_MD5: dir=out
src=6.25.232.1 iv_bits=64bits iv=0x7ba1ee06ec8458fe ooowin=64
alen=128 aklen=128 eklen=192 life(c,s,h)=addtime(45,0,0)
refcount=4 ref=15
```

- A section whose output is very similar to the **netstat –nr** section above.

Other causes for failure could include having:

- A firewall blocking traffic
- A firewall NAT-ing traffic. This is bad, please see the Chapter introduction.
- bad cables

Test The VPN Connectivity

You can test the VPN connectivity by sending a simple “ping” from one private network to the other. In this case we’re sending from the Windows server 10.0.0.105 protected by “vpn2” to server 172.16.1.1 which is protected by “vpn1”.

```
C:\>ping 172.16.1.1
Pinging 172.16.1.1 with 32 bytes of data:
Reply from 172.16.1.1: bytes=32 time=20ms TTL=253
Reply from 172.16.1.1: bytes=32 time<10ms TTL=253
```

```
Reply from 172.16.1.1: bytes=32 time=10ms TTL=253
Reply from 172.16.1.1: bytes=32 time<10ms TTL=253

Ping statistics for 172.16.1.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum =  20ms, Average =  7ms

C:\>
```

Protected Interface TCPDUMP Output From “vpn2”

Here we see unencrypted traffic successfully passing back and forth between the two servers.

```
23:46:46.536831 10.0.0.105 > 172.16.1.1: icmp: echo request
23:46:46.544844 172.16.1.1 > 10.0.0.105: icmp: echo reply
23:46:47.536292 10.0.0.105 > 172.16.1.1: icmp: echo request
23:46:47.543361 172.16.1.1 > 10.0.0.105: icmp: echo reply
```

Unprotected Interface TCPDUMP Output From “vpn2”

Here we see encrypted ESP traffic which is encapsulating the “pings” passing back and forth between the two VPN boxes. The true source and destination IP addresses (10.0.0.105 and 172.16.1.1) are hidden.

```
00:00:31.665730 vpn2.site2.com > vpn1.site1.com:
ESP(spi=0xcbf056ff,seq=0x9)
00:00:31.668408 vpn2.site2.com > vpn1.site1.com:
ESP(spi=0xcbf056ff,seq=0x9)
00:00:31.672554 vpn1.site1.com > vpn2.site2.com:
ESP(spi=0x2bc459c8,seq=0x9)
00:00:31.673793 vpn1.site1.com > vpn2.site2.com:
ESP(spi=0x2bc459c8,seq=0x9)
```

Possible Changes To IP Tables NAT/Masquerade Rules

If you are running iptables with masquerading/NAT the VPN devices then you will have to exclude packets traversing the tunnel from the NAT operation. This example assumes that interface **eth0** is the Internet facing interface on your Linux VPN/firewall.

Left Hand Side VPN Device

Old

```
iptables -t nat -A POSTROUTING -o eth0 -s 172.168.1.0/24 -j
MASQUERADE
```

New

```
iptables -t nat -A POSTROUTING -o eth0 -s 172.168.1.0/24 -d \!
10.0.0.0/24 -j MASQUERADE
```

Right Hand Side VPN Device

Old

```
iptables -t nat -A POSTROUTING -o eth0 -s 10.0.0.0/24 -j
MASQUERADE
```

New

```
iptables -t nat -A POSTROUTING -o eth0 -s 10.0.0.0/24 -d \!
176.16.1.0/24 -j MASQUERADE
```

How To Ensure FreeS/WAN Starts When Rebooting

The “sample” subsection in **/etc/ipsec.conf** will have a line:

```
auto=add
```

This will only authorize ipsec, but won’t establish the connection at startup. You’ll need to change this in the “real” section to:

```
auto=start
```

Once this is done, ipsec will start automatically on rebooting.

Using Pre-Shared Keys (PSK)

Create The PSK

- You can create a random pre-shared key by a key using the **ipsec** command below:

```
[root@vpn2 tmp]# ipsec ranbits --continuous 128
0x33893a081b34d32a362a46c404ca32d8
[root@vpn2 tmp]#
```

- You can also create them out of your head. It is best to make them long (over 20 bytes). We’ll use this one from now on.

```
nonebutourselvescanfreeourminds
```

Update /etc/ipsec.secrets

You then have to add text in the following format at the beginning of the **/etc/ipsec.secrets** file

```
vpn1-ip-address vpn2-ip-address : PSK “key in quotations”
```

So in our example it would be:

```
97.158.253.25 6.25.232.6 : PSK "nonebutourselvescanfreeourminds"
```

Update /etc/ipsec.conf

The PSK configuration is very similar to the RSA configuration with exception that the **leftid**, **rightid**, **leftrsasigkey** and **rightrsasigkey** fields are omitted from the relevant "conn" subsection. You also need to add the **authtype=secret** command to the configuration.

```
conn net-to-net
    authby=secret                 # Key exchange method
    left=97.158.253.25            # Public Internet IP address of the
                                  # LEFT VPN device
    leftsubnet=172.16.1.0/24      # Subnet protected by the LEFT VPN
device
    leftnexthop=%defaultroute     # correct in many situations
    right=6.25.232.1              # Public Internet IP address of
                                  # the RIGHT VPN device
    rightsubnet=10.0.0.0/24       # Subnet protected by the RIGHT VPN
device
    rightnexthop=97.158.253.25    # correct in many situations
    auto=start                    # authorizes and starts this connection
                                  # on booting
```

Remember to have the same configuration on the Linux VPN boxes on either side of the tunnel.

Restart FreeS/WAN

You'll then have to restart FreeS/WAN to activate the new settings.