CHAPTER – 8

SETTING UP DHCP AND NIS

Overview:

If our business, organization, or home network has more than a few computers, administering each computer individually can be difficult. Renaming our domain or getting a new pool of IP addresses can result in changing configuration files on every computer on the network. A new member in our organization could mean adding a new user account to multiple computers.

Red Hat Linux offers several mechanisms for centrally configuring and distributing critical information associated with our network, its servers, and the people that use our computing resources. DHCP provides a means of dynamically configuring the IP addresses, network numbers, and server locations for the computers on our local network. NIS offers a means of distributing a variety of configuration files (containing such information as user accounts, passwords, and network addresses) to other Linux and UNIX systems on our network.

Introduction to DHCP:

DHCP stands for dynamic host configuration protocol. What it does is dynamically assign network settings from a server. In other words, instead of having to configure the parameters related to how our computer communicates with a network, it happens automatically.

Assigning an IP address dynamically is the most basic piece but there is a lot more to DHCP. This includes the netmask, host name, domain name, gateway and name servers. In addition, DHCP can supply other information such as a time server.

Many people are anti-DHCP, because they see it as a way that an ISP offers us an IP address that changes. This, of course, makes it difficult to advertise a server. On the other hand, DHCP can save a lot of ongoing configuration work within our company or organization. Besides the ISP provided DHCP servers, they commonly exist in inexpensive router boxes. Netgear, Linksys and other vendors offer these systems with multiple LAN ports, an 802.11b wireless interface or both. The Netgear RP114 is an example of a wired LAN, while the Linksys WAP11 is an 802.11b type. Many other product choices are available. When we use one, the router box becomes the system that the ISP knows about, and all of our real computers hide behind this box.

Hide? Effectively, yes. What is visible to the public Internet is the router. The LAN has private IP addresses and uses network address translation (NAT) to handle connections from the internal systems to the Internet. Although this isn't really a firewall, NAT offers a basic level of protection.

Most routers in this class allow us to:

- Clone the MAC (hardware) address of one of our computers. This allows us to make the ISP think it is talking to a computer system we previously identified rather than to a router with possibly multiple machines connected to it.

- Handle static IP addresses. This means we could pick a local network address (192.168.1.*x*, for example) and assign specific addresses in this range.
- Dynamically assign IP addresses from a specified range. For example, the router could be configured to offer DHCP for 20 different addresses, say 192.168.1.100 thru 192.168.1.119.

That is the basics of "DHCP for Beginners". If we simply are trying to decide between using DHCP or a static IP address, this may be enough information. On the other hand, we could decide to run a DHCP server on a Linux system. In that case, there are more options.

Setting up DHCP Server:

Assuming we have already set up the physical connections between our DHCP server and the client computers on our network (presumably an Ethernet LAN), the minimum tools we need to get the DHCP server working are:

- A firewall that allows DHCP access
- A configured /etc/dhcpd.conf file
- A running dhcpd server daemon (which can be started at boot time)

After the DHCP server is running, it broadcasts its availability as a DHCP server to the LAN. A client simply boots up (with an Ethernet network interface turned on and DHCP identified as its method of getting network addresses), and the information it needs to get up and running on the network is fed to it from the server.

Note: *The dhcpd.conf file allows an extraordinary amount of flexibility. To see the full set of options and parameters we can set in that file, refer to the dhcp-options and dhcpd.conf man pages (type man dhcp-options).*

☑ Opening Our Firewall for DHCP:

The firewall on our DHCP server must be configured to allow access to UDP ports 67 and 68. If we are using iptables (and we did not open ports 67 and 68 during installation), we can add a new rule to iptables and then save the changes permanently. Type the following as root:

```
# iptables -I INPUT -I eth0 -p udp --sport 67:68 --dport 67:68 -j ACCEPT
```

In this example, requests are allowed to and from ports 67 and 68 on the eth0 interface (which is our first Ethernet card). If our DHCP server is also a routing firewall for our network, we want to make sure that we are only offering DHCP services to our LAN and not to the Internet. (We need to figure out if eth0, eth1, or some other card is connected to our LAN.)

If the rule was accepted (type **iptables -L** to make sure), we can save our entire firewall configuration so that the new rule is included permanently. To do that, type the following (as root user):

```
# iptables-save > /etc/sysconfig/iptables
```

This updates our /etc/sysconfig/iptables file so that all the current rules (including the one we just added) are included the next time iptables is restarted.

☑ Configuring The dhcpd.conf File

Suppose we have a single pool of IP addresses that we want to distribute to a set of computers that are all on the same subnetwork. In other words, all the computers are connected to one hub (or a set of daisy-chained hubs). Here is an example of a simple dhcpd.conf file:

```
ddns-update-style interim;
ignore client-updates;

subnet 10.0.0.0 netmask 255.0.0.0 {

     option routers                10.0.0.1;
     option domain-name-servers     10.0.0.1;
     option subnet-mask           255.0.0.0;
     option domain-name           "handsonhistory.com";

     range dynamic-bootp 10.0.0.150 10.0.0.225;
     default-lease-time 21600;
     max-lease-time 43200;

     # Set name server to appear at a fixed address
     host ns {
          next-server ns1.handsonhistory.com;
          hardware ethernet 00:D0:B3:79:B5:35;
          fixed-address 10.0.0.1;
     }
}
```

In this example, this DHCP server is providing IP addresses for client computers on a small LAN. The first two lines tell the DHCP server not to update DNS records for the local domain based on the IP addresses it assigns.

The DHCP server is serving a single LAN: 10.0.0.0 network with a 255.0.0.0 netmask. Other data in this file define what information the DHCP server will hand out to clients on this LAN.

A single server at address 10.0.0.1 is used as the router (or gateway) and DNS server for the LAN. To ensure that this server always gets the fixed address of 10.0.0.1, a host entry is set to the hardware address (00:D0:B3:79:B5:35) for the Ethernet card on the host named ns.

The pool of addresses handed out by this DHCP server is 10.0.0.150 to 10.0.0.225, as set by the range dynamic-bootp line. (Using dynamic-bootp allows bootp and dhcp clients to get addresses.) Along with the IP address that each client is assigned, the client is also given the associated subnet-mask and domain name.

The IP addresses that the DHCP server hands out are leased to each client for a particular time. The default-lease-time (set to 21,600 seconds here, or six hours) is the time assigned if the client doesn't request a particular lease period. The max-lease-time (43,200 seconds here, or 12 hours) is the highest amount of time the server will assign, if the client requests it. Clients can renew leases, so they don't have to lose the IP address while they are still using it.

Expanding the dhcpd.conf File:

As we noted earlier, this is a very simple example that works well for a single network of client computers. Below are some examples of ways that we can expand our dhcpd.conf file.

- If we have multiple ranges of addresses on the same sub-network, we can add multiple range options to a subnet declaration. Here is an example:

```
subnet 10.0.0.0 netmask 255.0.0.0 {
range 10.0.0.10 10.0.0.100;
range 10.0.0.200 10.0.0.250;
}
```

This example causes the DHCP server to assign IP addresses between the ranges of 0.0.10 and 0.0.100 and between 0.0.200 and 0.0.250 on network number 10.

- We can set fixed addresses for particular host computers. In particular, we would want to do this for our server computers so that their addresses don't change. One way to do this is based on the Ethernet hardware address of the server's Ethernet card. All information for that computer can be contained in a host definition, such as the following:

```
host pine {
hardware ethernet 00:04:5A:4F:8E:47;
fixed-address 10.0.0.254;
}
```

Here, when the DHCP server encounters the Ethernet address, the fixed-address (10.0.0.254) is assigned to it. Type ifconfig -a on the server computer to see the address of its Ethernet hardware (while the interface is up). Within this host definition, we can add other options as well. For example, we could set the location of different routes (routersoption).

- Many of the options let us define the locations of various server types. These options can be set globally or within particular host or subnet definitions. For example:

```
option netbios-name-servers 10.0.0.252;
option time-servers 10.0.0.253;
```

In these examples, the netbios-name-servers option defines the location of the WINS server (if we are doing Windows file and print server sharing using Samba). The time-servers option sets the location of a time server on our network.

- The DHCP server can be used to provide the information an X Terminal or diskless workstation could use to boot up on the network. The following is an example of a definition we could use to start such a computer on our network:

```
host maple {
filename "/dwboot/maple.nb";
hardware ethernet 00:04:5A:4F:8E:47;
fixed-address 10.0.0.150;
}
```

In the previous example, the boot file used by the diskless workstation from the DHCP server is located at /dwboot/maple.nb. The hardware ethernet value identifies the address of the Ethernet card on the client. The client's IP address is set to 10.0.0.150. All of those lines are contained within a host definition, where the host name is defined as maple. (See the Thin Clients heading in ***Table B*** for other options that may be useful for configuring thin clients.)

Adding Options:

There are dozens of options we can use in the /etc/dhcpd.conf file to pass information from the DHCP server to DHCP clients. Table A describes data types we can use for different options. Table B describes options that are available.

Table A: Data Types	
Data Types	**Description**
ip-address	Enter ip-address as either an IP address number (11.111.111.11) or a fully-qualified domain name (comp1.handsonhistory.com). To use a domain name, the name must be resolvable to an IP address number.
int32, int16, int8, uint32, uint16, uint8	Used to represent signed and unsigned 32-, 16-, and 8-bit integers, respectively.
"string"	Enter a string of characters, surrounded by double quotes.
Boolean	Enter true or false when a boolean value is required.
data-string	Enter a string of characters in quotes ("client1") or a hexadecimal series of octets (00:04:5A:4F:8E:47).

Table B: DHCP Options	
Options	**Descriptions**
Names, Addresses, and Time	
option host-name *string;*	Indicates the name that the client computer can use to identify itself. It can either be a simple host name (for example, pine) or a fully-qualified domain name (for example, pine.handsonhistory.com). We may use this in a host declaration, where a host computer is identified by an Ethernet address.
option domain-name *string;*	Identifies the default domain name the client should use to resolve DNS host names.

option *subnet-mask ip-address;*	Associates a subnetwork mask with an IP address. For example, option 255.0.0.0 10.0.0.1;.
option time-offset *int32;*	Indicates the offset (in seconds) from the Universal Time Coordinate (UTC). For example, a six-hour UTC offset is set as follows: option time-offset 21600;.
Servers and Routers	
option routers *ip-address* [, *ip-address*...];	Lists, in order of preference, one or more routers connected to the local subnetwork. The client may refer to this value as the gateway.
option domain-name-servers *ip-address* [, *ip-address*...];	Lists one or more Domain Name System (DNS) servers that the client can use to resolve names into IP addresses. List servers in the order in which they should be tried.
option time-servers *ip-address* [, *ip-address*...];	Lists, in order of preference, one or more time servers that can be used by the DHCP client.
option ien116-name-servers *ip-address* [, *ip-address*...];	Lists, in order of preference, one or more IEN 116 name servers that can be used by the client. (IEN 116 name servers predate modern DNS servers and are considered obsolete.)
option log-servers *ip-address* [, *ip-address*...];	Lists one or more MIT-LCS UDP log servers. List servers in the order in which they should be tried.
option cookie-servers *ip-address* [, *ip-address*...];	Lists one or more Quote of the Day (cookie) servers (see RFC 865). List servers in the order in which they should be tried.
option lpr-servers ip-address [, ip-address...];	Lists one or more line printer servers that are available. List servers in the order in which they should be tried.
option impress-servers *ip-address* [, *ip-address*...];	Lists one or more Imagen Impress image servers. List servers in the order in which they should be tried.
option resource-location-servers *ip-address* [, *ip-address*...];	Lists one or more Resource Location servers (RFC 887). List servers in the order in which they should be tried.
option nis-domain *string;*	Indicates the name of the NIS domain, if an NIS server is available to the client.
option nis-servers *ip-address* [, *ip-address*...];	Lists addresses of NIS servers available to the client, in order of preference.
option ntp-servers *ip-address* [, *ip-address*...];	Lists addresses of network time protocol servers, in order of preference.
option netbios-name-servers *ip-address* [, *ip-address*...];	Lists the addresses of WINS servers, used for NetBIOS name resolution (for Windows file and print sharing).
option netbios-dd-server *ip-address* [, *ip-address*...];	Lists the addresses of NetBIOS datagram distribution (NBDD) servers, in preference order.
option netbios-node-type *uint8;*	Contains a number (a single octet) that indicates how NetBIOS names are determined (used with NetBIOS over TCP/IP). Acceptable values include: 1 (broadcast: no WINS), 2 (peer: WINS only), 4 (mixed: broadcast, then WINS), 8 (hybrid: WINS, then broadcast).

option font-servers *ip-address* [, *ip-address*...];	Indicates the location of one or more X Window font servers that can be used by the client, listed in preference order.
option nisplus-domain *string;*	Indicates the NIS domain name for the NIS+ domain.
option nisplus-servers *ip-address* [, *ip-address*...];	Lists addresses of NIS+ servers available to the client, in order of preference.
option smtp-server *ip-address* [, *ip-address*...];	Lists addresses of SMTP servers available to the client, in order of preference.
option pop-server *ip-address* [, *ip-address*...];	Lists addresses of POP3 servers available to the client, in order of preference.
option nntp-server *ip-address* [, *ip-address*...];	Lists addresses of NNTP servers available to the client, in order of preference.
option www-server *ip-address* [, *ip-address*...];	Lists addresses of WWW servers available to the client, in order of preference.
option finger-server *ip-address* [, *ip-address*...];	Lists addresses of Finger servers available to the client, in order of preference.
option irc-server *ip-address* [, *ip-address*...];	Lists addresses of IRC servers available to the client, in order of preference.
Routing	
option ip-forwarding *flag;*	Indicates whether the client should allow (1) or not allow (0) IP forwarding. This would be allowed if the client were acting as a router.
option non-local-source-routing *flag;*	Indicates whether or not the client should allow (1) or disallow (0) datagrams with nonlocal source routes to be forwarded.
option static-routes *ip-address ip-address* [, *ip-addressip-address*...];	Specifies static routes that the client should use to reach specific hosts. (List multiple routes to the same location in descending priority order.)
option router-discovery *flag;*	Indicates whether the client should try to discover routers (1) or not (0) using the router discovery mechanism.
option router-solicitation-address *ip-address;*	Indicates an address the client should use to transmit router solicitation requests.
Thin Clients	
option boot-size *uint16;*	Indicates the size of the default boot image (in 512-octet blocks) that the client computer uses to boot.
option merit-dump *string;*	Indicates where the core image should be dumped if the client crashes.
option swap-server *ip-address;*	Indicates where the client computer's swap server is located.
option root-path *string;*	Indicates the location (path name) of the root disk used by the client.
option tftp-server-name *string;*	Indicates the name of the TFTP server that the client should use to transfer the boot image. Used more often with DHCP clients than with BOOTP clients.

option bootfile-name *string;*	Indicates the location of the bootstrap file that is used to boot the client. Used more often with DHCP clients than with BOOTP clients.
option x-display-manager *ip-address* [, *ip-address*...];	Indicates the locations of X Window System Display Manager servers that the client can use, in order of preference.

Options contain values that are passed from the DHCP server to clients. Although Table B lists valid options, the client computer will not be able to use every value we could potentially pass to it. In other words, not all options are appropriate in all cases.

Table B is divided into the following categories:

- **Names, Addresses, and Time**: These options set values that are used by clients to have their host name, domain name, network numbers, and time (offset from GMT) defined.
- **Servers and Routers**: These options are used to tell DHCP clients where on the network to find routers and servers. Though more than a dozen server types are listed, most often you will just indicate the address of the router and the DNS servers the client will use.
- **Routing**: These options indicate whether or not the client routes packets.
- **Thin Clients**: These options are useful if DHCP is being used as a boot server for thin clients. A thin client may be an X Terminal or diskless workstation that has processing power, but no disk (or a very small disk) so it can't store a boot image and a file system itself.

☑ Starting The DHCP Server:

After the /etc/dhcpd.conf file is configured, we can start the DHCP server immediately. As root user from a Terminal window, type the following:

```
# service dhcpd start
```

Our DHCP server should now be available to distribute information to the computers on our LAN. If there are client computers on our LAN waiting on our DHCP server, their network interfaces should now be active.

If everything is working properly, we can have our DHCP server start automatically each time our computer boots by turning on the dhcpd service as follows:

```
# chkconfig dhcpd on
```

There are a few ways we can check that our DHCP server is working:

Check the /var/lib/dhcp/dhcpd.leases file. If a client has successfully been assigned addresses from the DHCP server, a lease line should appear in that file. There should be one set of information that looks like the following for each client that has leased an IP address:

```
lease 10.0.0.225 {
        starts 2 2002/05/04 03:48:12;
```

```
        ends 2 2002/05/04 15:48:12;
        hardware ethernet 00:50:ba:d8:03:9e;
        client-hostname "pine:;
}
```

Turn on the Ethereal window (type ethereal& from a Terminal window) and start capturing data (in promiscuous mode). Restart the DHCP server and restart the network interface on the client. We should see a series of DHCP packets that show a sequence that looks like the following: DHCP Offer, DHCP Discover, DHCP Offer, DHCP Request, and DHCP ACK.

From the client computer, we should be able to start communicating on the network. If the client is a Linux system, type the ifconfig -a command. Our Ethernet interface (probably eth0) should appear, with the IP address set to the address assigned by the DHCP server.

When the server is running properly, we can continue to add DHCP clients to our network to draw on the pool of addresses we assign.

Setting Up a DHCP Client:

Configuring a network client to get addresses from our DHCP server is fairly easy. Different types of operating systems, however, have different ways of using DHCP. Here are examples for setting up Windows and Red Hat Linux DHCP clients.

Windows:

1. From most Windows operating systems (Windows 95, 98, 2000, ME, and so on), we open the Network window from the Control Panel (Start>Settings>Control Panel).
2. From the Configuration tab, click the TCP/IP interface associated with our Ethernet card (something like TCP/IP ? 3Com EtherLink III).
3. Click Properties. The Properties window appears.
4. Click the IP Address tab and then select "Obtain an IP Address Automatically".
5. Click OK and reboot the computer so the client can pick up the new IP address.

Red Hat Linux:

1. While we are initially installing Red Hat Linux, click Configure using DHCP on the Network Configuration screen. Our network client should automatically pick up its IP address from our DHCP server when it starts up.
2. To set up DHCP after installation, open the Network Configuration window (neat command).
3. From the Network Configuration window:
   a. Click the Devices tab (on by default).
   b. Click Ethernet device (probably eth0).
   c. Click Edit.
   d. Click the General tab.
   e. Click "Automatically obtain IP address Settings with" and select dhcp.
   f. Select OK.
   g. Select Apply.

4. Then, from a Terminal window, type:
5. # **/etc/init.d/network restart**

By default, a Red Hat Linux client will not accept all information passed to it from the DHCP server. The way that the Red Hat client handles DHCP server input is based on settings in the /etc/sysconfig/network-scripts/ifup script. If the client has DHCP turned on, when the system starts up networking, the ifup script runs the dhcpcd command as follows:

- If the dhcpcd client process is currently running, the dhcpcd command sends a signal to it so that it asks the DHCP server to renew the lease on the IP address.
- If no host name is set on the client (or if the host name is set to localhost), the -H option is passed to dhcpcd to indicate that it should accept the host name supplied by the DHCP server. (If the host name is already set, the client will not reset the host name from the server.)
- Any new DNS server assignments are accepted by the client. If your DNS servers are already configured in the /etc/resolv.conf file, then you can have the -R option passed to dhcpcd to prevent it from updating that file with new DNS server information. (To do this, add PEERDNS=no to the /etc/sysconfig/network file on the client.)

To change how the dhcpcd command works to accept information from the DHCP server, we can pass options to the dhcpcdcommand. Do this by adding arguments to the DHCPCDARGS variable in the /etc/sysconfig/network configuration file. For example, DHCPCDARGS="-d" causes the ifup script to run dhcpcd in debug mode so that messages are sent to the /var/log/messages file. (Type **man dhcpcd** to see other dhcpcd options.)

The dhcpcd daemon assumes that our DHCP server is accessible on our first Ethernet interface (eth0). If we want the client to get its DHCP information from a different interface, we can add the interface name (eth1, eth2, etc.) to the dhcpd command line. For example, to do this we could add DHCPCDARGS=eth1 to the /etc/sysconfig/network file.

Understanding Network Information Service (NIS):

Network Information Service (NIS) was created by Sun Microsystems as a way of managing information that is shared among a group of host computers on a network. Using NIS, computers can share a common set of user accounts, user groups, and TCP/IP host names, as well as other information.

Note: *NIS was originally called Yellow Pages, but Sun had to change this name because it was trademarked. Some people still refer to NIS as YP, and many of the NIS commands (and even NIS package names) begin with the letters "yp." To use NIS as a client, we need to have the ypbind and yp-tools packages installed. To configure an NIS server, you need the ypserv package installed as well.*

The information we share with NIS comes from files that are used with UNIX systems and, therefore, compatible with other UNIX-like systems, such as Red Hat Linux. The group of computers that the master NIS server supports is referred to as an *NIS domain*. This domain is a defined set of host computers that may or may not be the same group of computers contained in a TCP/IP domain.

With NIS, an administrator creates information databases called *maps* from common UNIX (or Linux) system files. The NIS maps are created on the master NIS server and are accessible to other host computers from that server. Just in case the master server is down or inaccessible, one or more slave servers can be defined. The NIS slave servers contain copies of the NIS maps and can provide that information to client computers when the master is unavailable. However, NIS slave servers are not used to create the maps.

When the maps have been shared among the computers in the NIS domain, the main result is that all the computers share a common set of users and network configuration. The following is a list of files that are available for sharing by NIS (not all of them are set up for sharing by default).

- **/etc/group**: Defines the groups to which users on the computer belong.
- **/etc/passwd**: Defines the users who have accounts set up on the computer.
- **/etc/shadow**: Contains encrypted passwords for the users set up in the /etc/passwd file.
- **/etc/gshadow**: Contains encrypted passwords associated with groups contained in the /etc/groups file. (This file is optional and is usually not used.)
- **/etc/passwd.adjunct**: Secures password entries if our system doesn't use shadow passwords. (This file is used with SunOS systems.)
- **/etc/aliases**: Contains user aliases used with e-mail. It allows mail that is sent to a particular user name to be directed to a different user (or set of users). On some systems, this file may be /etc/mailaliases instead.
- **/etc/ethers**: Used by the RARP to map Ethernet addresses into IP numbers. This file is optional. (By default, RARP support is not configured into Red Hat Linux.)
- **/etc/bootparams**: Contains entries needed to start diskless workstations (typically used to boot Sun Microsystems diskless workstations).
- **/etc/hosts**: Contains the names and IP addresses of computers that can be reached on TCP/IP networks. (Often used to contain all the addresses for a private LAN, while Internet addresses would be determined from a DNS server.)
- **/etc/networks**: Used to attach a name to a network. In this way, we can refer to networks by name rather than by number.
- **/etc/printcap**: Contains printer definitions.
- **/etc/protocols**: Identifies numbers that are assigned to different Internet network protocols (such as IP, TCP, UDP, and others).
- **/etc/publickey**: Used on some UNIX systems to contain user names and associated public and private keys for secure networking in NFS and related features.
- **/etc/rpc**: Contains listings of supported Remote Procedure Call (rpc) protocols. These protocols are used with Sun Microsystems UNIX systems to allow requests for network services, such as NIS and others.
- **/etc/services**: Contains listings that identify port number and protocols for supported network services that are used with Internet protocols.
- **/etc/netgroup**: Used to define users (from particular hosts and domains) for permission-checking associated with remote mounts, remote shells, and remote logins.
- **/etc/netid**: Contains information that maps RPC network names to UNIX credentials.

Note: *Some of the files just shown may not be applicable to our Red Hat Linux system. Don't worry if some of these files don't exist. In the course of setting up our system (adding users, configuring networks, and so on), we will set up the files we need.*

Although these files are created in the /etc directory, the NIS administrator can copy these files to a different location and change them, so as not to share the master NIS server's original configuration files. Files can also be added to this list or removed from the list as the NIS administrator chooses. When an NIS client computer is configured, this configuration information can be obtained from the NIS master server.

Setting Up Red Hat Linux as an NIS Client:

If our network uses NIS centrally to administer users, groups, network addresses, and other information, we can set up our Red Hat Linux system to use that information as an NIS client. To configure Red Hat Linux as an NIS client, we need to get the following information from our NIS administrator:

- **NIS Domain Name**:

This is a keyword used to describe the group of hosts that use the common set of NIS files. Domain name is an unfortunate way of referring to this keyword, because it doesn't have anything to do with the TCP/IP domain name. Its only similarity is that it refers to a group of computers.

- **NIS Master Server Name**:

This is the name of the computer on our network that maintains the NIS databases and responds to requests from the network for that information.

- **NIS Slave Server Names**:

An NIS domain may have more than one NIS server that can handle requests for information from the domain's NIS database. An NIS slave server keeps copies of the NIS maps so that it can respond to requests if the master NIS server goes down. (NIS slave servers are optional.)

When we installed Red Hat Linux, if we knew that our network used NIS, we could have selected NIS as the way to handle user names and passwords on our computer. If we have not already configured NIS for our computer, the procedures that follow will describe how to do that. The procedures consist of defining our NIS domain name, setting up the /etc/yp.conf file, and configuring NIS client daemons (ypbind and ypwhich) to start when we boot our system.

☑ Defining An NIS Domain Name:

We can set our Red Hat Linux computer's NIS domain name using the domain name command. For example, if our NIS domain name were trident, we could set it by typing the following as the root user from the shell:

```
# domainname trident
```

To verify that our NIS domain name is set, simply type domainname and we will see the name. Unfortunately, we're not done yet. Running domainname doesn't set the NIS domain name permanently. As soon as we reboot the computer, it is gone. (we can verify this by typing domainname again after a reboot.)

To make the NIS domain name permanent, we need to have the domainname command run automatically each time our system boots. There are many ways to do this. What we did was add the command line (domainname trident) to a run-level script that runs before the ypbind daemon is started. We edited the /etc/init.d/network file and added the following lines just after the first set of comment lines (about line number 9).

```
# Set the NIS domain name.
domainname trident
```

This caused my NIS domain name to be set each time my Red Hat Linux system booted. When we add this entry, make sure we spell the NIS domain name properly (including upper- and lowercase letters). If we get it slightly wrong, we will see ypbind failure messages when we boot.

Caution: *Be very careful editing a run-level script. Make a copy before we edit it. If we make a mistake editing one of these files, we could find ourself with a network or other essential service that doesn't work. we also risk losing this information when we upgrade our system at a later date.*

☑ Setting up the /etc/yp.conf file:

The ypbind daemon needs information about our NIS domain and NIS servers for it to work. That information is set up in our /etc/yp.conf file. The first entries define our NIS domain name and NIS servers. For example, if we had an NIS domain called trident and a master server called maple, we would have the following entry in our /etc/yp.conf file:

```
domain trident server maple
```

If we had other slave NIS servers named oak and pine, for example, we could also have the following entries:

```
domain trident server oak
domain trident server pine
```

We can also set our computer to broadcast to the local network for our NIS server. If our domain were named trident, for example, we would use the domain/broadcast option as follows:

```
domain trident broadcast
```

If the address of our NIS server is contained in our /etc/hosts file, we can specify that ypbind look in that file to find the server's IP address. For example, if our NIS master server is named maple, we would add the following entry:

```
ypserver maple
```

When ypbind starts, all the information in this file is read. It is then used to contact the appropriate NIS server.

☑ Configuring NIS client daemons:

After our NIS client information is all set up, all we need to do to run NIS as a client is start the ypbind and ypwhichdaemons. The ypbind daemon runs continuously as two processes: The

master ypbind process handles requests for information from our NIS server, and the slave ypbind process checks the bindings from time to time. The ypwhich daemon finds our NIS master server.

Getting these daemons running is pretty easy. We can set up an existing run-level script called ypbind to start automatically at boot time. To do this, we can run the following command (as root user from a Terminal window):

```
# chkconfig ypbind on
```

To start the ypbind daemon immediately, type:

```
# /etc/init.d/ypbind start
```

☑ Checking that NIS is working:

To check that our NIS client is communicating with our NIS master server, follow the instructions in this section.

Note: *If our NIS server isn't configured yet, refer to the "Setting Up Red Hat Linux as an NIS Master Server" section later in this chapter to configure our NIS server. Then return to this procedure to make sure that everything is working properly.*

From the NIS client computer, type the following command to make sure that we are communicating with the NIS server:

```
# ypwhich
maple
```

The output shown above indicates that the NIS client is bound to the NIS server named *maple*. Next, check that the maps are being shared using the ypcat command. (To see what files are being shared from the NIS server, look in the server's /var/yp/nisdomain directory, where *nisdomain* is replaced by our NIS domain name.) Type one of the files shown in that directory along with the ypcat command. Here's an example:

```
# ypcat hosts
10.0.0.45      ash
10.0.0.46      pine
10.0.0.47      maple
```

If we are communicating with the NIS server and able to access map files, we can now define which maps the NIS client uses of those shared map files.

☑ Using NIS maps:

For the information being distributed by the NIS server to be used by the NIS client, we must configure the /etc/nsswitch.conffile to include nis in the search path for each file we want to use.

The following is a listing from the /etc/nsswitch.conf file showing valid values that can be in the search paths for accessing different configuration files.

```
# Legal entries are:
#
#   nisplus or nis+    Use NIS+ (NIS version 3)
#   nis or yp          Use NIS (NIS version 2), also called YP
#   dns                Use DNS (Domain Name Service)
#   files              Use the local files
#   db                 Use the local database (.db) files
#   compat             Use NIS on compat mode
#   hesiod             Use Hesiod for user lookups
#   [NOTFOUND=return]  Stop searching if not found so far
#
```

For our purposes, we want to add nis into the paths for the files we want to distribute from our NIS server to this NIS client. In most cases, only the local files are checked (files). The following are examples of how some entries appear:

```
passwd:     files
shadow:     files
group:      files
hosts:      files dns
```

For each of these entries, the original files are checked first (/etc/passwd, /etc/shadow, and so on). For host names, the DNS server is checked after the local hosts file. For our purposes, we can add nis to access the maps being shared from the NIS server. (Linux NIS servers only implement nis and not nisplus.) The lines would then appear as follows:

```
passwd:     files nis
shadow:     files nis
group:      files nis
hosts:      files nis dns
```

As soon as the /etc/nsswitch file is changed, the data from the NIS maps are accessible. No need to restart the NIS service. We can go through and change any of the files listed in the /etc/nsswitch file so that it is configured to let our system access the NIS maps being shared.

Setting Up Red Hat Linux as an NIS Master Server:

To configure our Red Hat Linux system as an NIS master server, we should first configure it as an NIS client (That is, set the NIS domain name, set up /etc/yp.conf, and configure client daemons as described earlier.) Then we create the NIS maps and configure the NIS master server daemon processes (ypserv and rpc.yppasswdd).

Note: *If there is a firewall on our NIS server, we must make UDP port 111 (sunrpc) available or NIS clients won't be able to connect to our NIS service. If the computer is also a router, if possible, block access to port 111 outside of our local network.*

☑ Creating NIS Maps:

To create NIS maps so that our Red Hat Linux system can be an NIS master server, start from the /var/yp directory from a Terminal window as root user. In that directory, a Makefile enables us to configure which files are being shared with NIS. The files that are shared by default are listed near the beginning of this chapter and within the Makefile itself.

☑ Choosing Files To Map

If we don't want to share any file that is set up in the Makefile, we can prevent that file from being built. Do this by finding the following line in the Makefile and simply deleting the file we want excluded:

```
all: passwd group hosts rpc services netid protocols mail \
       # netgrp shadow publickey networks ethers bootparams printcap \
       # amd.home auto.master auto.home auto.local passwd.adjunct \
       # timezone locale netmasks
```

We may notice that not all the names in the all: line represent the exact filename. For example, netgrp is for the /etc/netgroupfile. The files that each name represents are listed a few lines below the all: line in the Makefile. We may also notice that many of the files are already commented out, including the shadow file.

Tip: *The NIS-HOWTO document suggests that using shadow passwords with NIS is "always a bad idea." Options in the Makefile (described in the next section) enable us to automatically merge the shadow and gshadow files into the passwd and group files, respectively.*

☑ Choosing Mapping Options:

Within the Makefile, several options are set. We can choose to change these options or leave them as they are. Here are the options:

- **B=**: We can use the B= option to allow NIS to use the domain name resolver to find hosts that are not in the current domain. By default, B= is not set. To turn on this feature, set it to -b (B=-b).
- **NOPUSH=true**: When set to true (the default), the NOPUSH option prevents the maps from being pushed to a slave server. This implies that the NIS master server is the only server for the NIS domain. Set this to false, and place the host names of slave servers into the /var/yp/ypservers file if we do not want the NIS master to be the only server for the domain.
- **MINUID=500:** To prevent password entries from being distributed for administrative users, the MINUID is set to 500. This assumes that all regular user accounts on the system that you want to share have UIDs that are 500 or above.

- **MINGID=500**: To prevent password entries from being distributed for administrative groups, the MINGID is set to 500. This assumes that all regular groups that we want to share have GIDs that are 500 or above.

Note: *Tools for adding users and groups to Red Hat Linux always begin with the number 500. Some UNIX and Linux systems, however, may use lower UIDs and GIDs for regular users. In those cases, we may need to lower MINUID and MINGID to below 500, but not below 100 (which always represent administrative logins).*

- **MERGE_PASSWD=true**: Keep this option true if we want each user's password to be merged from the shadow file back into the passwd file that is shared by NIS.
- **MERGE_GROUP=true**: Keep this option true if we want each group's password to be merged from the gshadow file back into the group file that is shared by NIS.

To build the NIS maps, our system must have the awk, make, and umask commands. In the Makefile, the locations of these commands are /usr/bin/gawk, /usr/bin/gmake, and umask, respectively. (The umask command is a shell built-in command, so we don't have to look for its location.) We can use comparable commands in different locations by changing the values of the AWK, MAKE, and UMASK variables in the Makefile.

Besides the options just mentioned, there are several variables we can set to change the location of NIS files. For example, the locations of password files (YPPWDDIR) and other source files (YPSRCDIR) are both set to /etc by default. The location of YP commands (YPBINDIR) is set to /usr/lib/yp. If we want to change the values of these or other variables, we can do so in the Makefile.

☑ Defining NIS Client Access:

Add the IP addresses of the client computers that are allowed access to our NIS maps to the /var/yp/securenets file. By default, any computer on any network that can reach our NIS master can have access to our maps (which is not a secure situation). So, it is important that we configure this file. IP numbers can be given in the form of netmask/network pairs. For example:

```
255.255.255.0           10.0.0.0
```

This example enables access to our NIS master server maps from all computers on network number 10.0.0.

☑ Configuring Access To Maps:

In the /etc/ypserv.conf file, we can define rules regarding which client host computers have access to which maps. We can also set several related options. Access rules in the ypserv.conf file have the following format:

```
host:map:security:mangle[:field]
```

Asterisks can replace *host* and *map* fields to create rules that match any host or map, respectively. The *host* is the IP address for the network or particular host for which the rule applies. The *map* is the name of the map for which we are defining access. The *security* is replaced by none (to always allow access), port (to allow access from a port less than port number 1024), deny (to deny access to this map), or des (to require DES authentication).

The *mangle* is replaced by yes or no (to indicate if a field in the map should be replaced by an x if a request comes from an unprivileged host). If the mangle is set to yes, *field* is replaced by the name of the field that should be mangled (the second field is used by default).

The following options can be set in the ypserv.conf file:

- **dns**: If yes (dns: yes), NIS will query the TCP/IP name server for host names when host names are not found in maps. By default, dns:no is set.
- **xfr_check_port**: If yes (xfr_check_port:yes), the NIS master server must run on a port that is less than port number 1024. If no, any port number may be used. By default, this is set to yes.

If we make changes to the /etc/ypserv.conf file, the ypserv daemon will pick up those changes the next time our system reboots (or the ypserv service restarts). Alternatively, we can have ypserv read the contents of the file immediately by sending the ypserv process a SIGHUP signal. Removing the comment character (#) from the following line in /etc/ypserv.confallows all hosts access to all maps:

```
* : * : * : none
```

☑ Generating The NIS Map Database

To install and build the NIS database, run the ypinit command. To start the ypinit program, type the following:

```
# /usr/lib/yp/ypinit -m
   next host to add:  maple
   next host to add:
```

The ypinit command should automatically choose our host name to use as an NIS server. After that, it asks us to add slave servers. Add one at a time; then press Ctrl+D after we have entered our last slave server. Verify that the list of NIS servers is correct (type **y**). (Remember that slave servers are not required.)

The database is built at this point. A new directory that has the name of our NIS domain is created in /var/yp. For example, if our NIS domain name is trident, the directory is /var/yp/trident. All maps built are then placed in that directory.

☑ Adding NIS Slave Servers:

In Red Hat Linux, NIS is configured by default to have a master NIS server and no slave NIS servers. We can allow our NIS maps to be pushed to one or more slave servers by setting NOPUSH=false in the /var/yp/Makefile file. After that, we need to add the names of the slave servers to our /var/yp/ypservers file. We can either add the host names manually or have them added automatically when we run the ypinit command later.

☑ Configuring NIS Server Daemons:

The NIS server must be running several daemon processes to be an NIS server. Red Hat Linux supplies run-level scripts that we can configure to start NIS server daemon processes. These scripts, located in the /etc/init.d directory, include the following:

- **ypserv:** This script starts the ypserv (/usr/sbin/ypserv) daemon. It reads information from the /etc/ypserv.conf file to determine what to do. Then it listens for requests from NIS client computers on the network.
- **yppasswdd**: This script starts the rpc.yppasswdd (/usr/sbin/rpc.yppasswdd) daemon. This daemon handles requests from users on NIS client computers who want to change their user passwords.

Unless we requested that these scripts be configured to start at boot time when we installed Red Hat Linux, they will not start automatically. We can use the following chkconfig command to set ypserv and yppasswdd scripts to start automatically at boot time.

```
# chkconfig ypserv on
# chkconfig yppasswdd on
```

If we want to start the services immediately, we can type the following:

```
# /etc/init.d/ypserv start
# /etc/init.d/yppasswdd start
```

The NIS master server should be up and running. If there are any NIS slave servers, we should configure them now.

Setting Up Red Hat Linux as an NIS Slave Server

To set up an NIS slave server, we must configure it as we do an NIS master server, but with one exception: Instead of creating the NIS maps, we run the ypinit command so that the NIS maps can be copied from the server. The option that we give to ypinit is the *-s master* option, where *master* is replaced by the name of our NIS master server. Here is an example of running ypinit where the NIS master server is named maple:

```
# /usr/lib/yp/ypinit -s maple
```

As long as the NIS slave server is allowed access, the maps should be copied to our computer from the NIS master server. If the NIS master server goes down, this slave computer should be able to handle NIS requests from the network.

At this point, we can return to the section on setting up NIS as a client to make sure that our NIS server is running properly and distributing the maps to its clients.